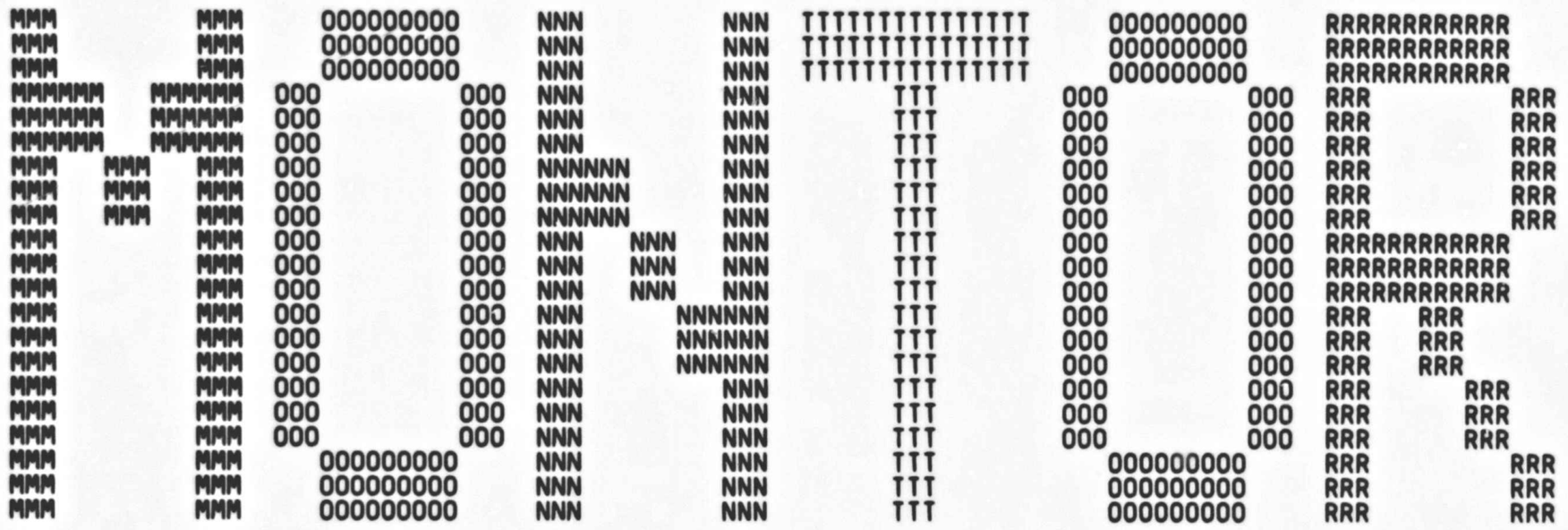
MMM MMM 000000000 NNN NNN TTTTTTTTTTTTTTT 000000000 RRRRRRRRRRR
MMM MMM 000000000 NNN NNN TTTTTTTTTTTTTTT 000000000 RRRRRRRRRRRR
MMM MMM 000000000 NNN NNN TTTTTTTTTTTTTTT 000000000 RRRRRRRRRRRR
MMMMMMM MMMMMMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMMMMMM MMMMMMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMMMMMM MMMMMMM 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**SUMMBUFF

```
  SSSSSSSS  UU      UU  MM      MM  MM      MM  BBBBBBBB    UU      UU  FFFFFFFFFF  FFFFFFFFFF
  SSSSSSSS  UU      UU  MM      MM  MM      MM  BBBBBBBB    UU      UU  FFFFFFFFFF  FFFFFFFFFF
SS          UU      UU  MMMM  MMMM  MMMM  MMMM  BB      BB  UU      UU  FF          FF
SS          UU      UU  MMMM  MMMM  MMMM  MMMM  BB      BB  UU      UU  FF          FF
SS          UU      UU  MM  MM  MM  MM  MM  MM  BB      BB  UU      UU  FF          FF
SS          UU      UU  MM  MM  MM  MM  MM  MM  BB      BB  UU      UU  FF          FF
  SSSSSS    UU      UU  MM      MM  MM      MM  BBBBBBBB    UU      UU  FFFFFFFF    FFFFFFFF
  SSSSSS    UU      UU  MM      MM  MM      MM  BBBBBBBB    UU      UU  FFFFFFFF    FFFFFFFF
        SS  UU      UU  MM      MM  MM      MM  BB      BB  UU      UU  FF          FF
        SS  UU      UU  MM      MM  MM      MM  BB      BB  UU      UU  FF          FF
        SS  UU      UU  MM      MM  MM      MM  BB      BB  UU      UU  FF          FF          ....
        SS  UU      UU  MM      MM  MM      MM  BB      BB  UU      UU  FF          FF          ....
SSSSSSSS    UUUUUUUUUU  MM      MM  MM      MM  BBBBBBBB    UUUUUUUUUU  FF          FF          ....
SSSSSSSS    UUUUUUUUUU  MM      MM  MM      MM  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) 69 DECLARATIONS
(4) 230 FILL_MFSUM_FAOSTK - Fill FAOSTK for M.F. Summary
(16) 662 CAPTURE_SUMS - Move SUM buffers to Summary Buffer
(18) 870 FILL_HOM_SUMMBUFF - Move SUM buffers to Summary Buffer (Homog)
(21) 1040 SUM_TO_SUMMBUFF - Move SUM buffer to Summary Buffer
(22) 1144 ALLOC_SUMBUFS - Allocate Summary Buffers
(24) 1297 MFS_FREE_MEM - Free Virtual Memory for m.f.s.

```
0000     1          .TITLE  SUMMBUFF - Multi-file Summary Buffer Routines
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;**********************************************************************************
0000     6  ;*                                                                                *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                        *
0000     9  ;*  ALL RIGHTS RESERVED.                                                          *
0000    10  ;*                                                                                *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED          *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER        *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY        *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
0000    16  ;*  TRANSFERRED.                                                                  *
0000    17  ;*                                                                                *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE         *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT        *
0000    20  ;*  CORPORATION.                                                                  *
0000    21  ;*                                                                                *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS        *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                       *
0000    24  ;*                                                                                *
0000    25  ;*                                                                                *
0000    26  ;**********************************************************************************
0000    27
0000    28  ;++
0000    29  ; FACILITY:  VAX/VMS MONITOR Utility
0000    30  ;
0000    31  ; ABSTRACT:
0000    32  ;
0000    33  ;       The routines in the SUMMBUFF module manipulate the multi-file
0000    34  ;       summary buffer for each class. This buffer is designed to
0000    35  ;       contain the accumulated sums of all counts and levels for
0000    36  ;       all elements for all input files.
0000    37  ;
0000    38  ; ENVIRONMENT:
0000    39  ;
0000    40  ;       User mode, IPL 0, unprivileged.
0000    41  ;
0000    42  ; AUTHOR: Thomas L. Cafarella, January, 1984
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;
0000    46  ;       V03-004 TLC1088         Thomas L. Cafarella     25-Jul-1984     14:00
0000    47  ;               Free virtual memory obtained for multi-file summary.
0000    48  ;
0000    49  ;       V03-003 TLC1071         Thomas L. Cafarella     17-Apr-1984     14:00
0000    50  ;               Add "number of input files" to multi-file summary report.
0000    51  ;
0000    52  ;       V03-002 TLC1074         Thomas L. Cafarella     04-May-1984     16:00
0000    53  ;               Correct XQPCACHE multi-file summary floating overflow.
0000    54  ;
0000    55  ;       V03-001 PRS1021         Paul R. Senn            12-Apr-1984     10:00
0000    56  ;               Make SYSTEM class work with multi-file summary.
0000    57  ;
```

```
0000     58 ;        V03-001 PRS1020         Paul R. Senn            12-Apr-1984     10:00
0000     59 ;                Make XQP class work with multi-file summary.
0000     60 ;
0000     61 ;        V03-001 TLC1061         Thomas L. Cafarella     18-Mar-1984     11:00
0000     62 ;                Identify dual-path disks by allocation class.
0000     63 ;
0000     64 ;        V03-001 TLC1060         Thomas L. Cafarella     12-Mar-1984     11:00
0000     65 ;                Make multi-file summary work for homogeneous classes.
0000     66 ;
0000     67 ;--
```

```
                        69          .SBTTL  DECLARATIONS
00000000                70          .PSECT  MONRODATA,QUAD,NOEXE,NOWRT
                        71 ;
                        72 ; INCLUDE FILES:
                        73 ;
                        74
                        75          $CDBDEF                         ; Define Class Descriptor Block
                        76          $CDXDEF                         ; Define CDB Extension
                        77          $CSBDEF                         ; Define Column Summary Block
                        78          $IDBDEF                         ; Define item descriptor block offsets
                        79          $MCADEF                         ; Define Monitor Communication Area
                        80          $MRBDEF                         ; Define Monitor Request Block
                        81          $MBPDEF                         ; Define Monitor Buffer Pointers
                        82          $MFSDEF                         ; Define Multi-File Summary Block
                        83          $TM2DEF                         ; Define temporary storage offsets
                        84          $TM3DEF                         ; Define temporary storage offsets
                        85
                        86 ;
                        87 ; MACROS:
                        88 ;
                        89
                        90 ;
                        91 ; Local Macro Definitions
                        92 ;
                        93
                        94 ;
                        95 ; ALLOC Macro - Dynamically allocate space on the stack.
                        96 ;
                        97
                        98          .MACRO  ALLOC   LENGTH,RSLDESC,RSLBUF
                        99          SUBL    #<LENGTH+3>&<^C3>,SP
                       100          .IF     NB,RSLBUF
                       101          MOVL    SP,RSLBUF
                       102          .ENDC
                       103          PUSHL   SP
                       104          PUSHL   #LENGTH
                       105          MOVL    SP,RSLDESC
                       106          .ENDM   ALLOC
                       107
                       108 ;
                       109 ; EQUATED SYMBOLS:
                       110 ;
                       111
0000001B               112 ESC = 27                                ; escape character
                       113
```

```
                                0000   115 ;
                                0000   116 ; OWN STORAGE (Read-only):
                                0000   117 ;
                                0000   118
                                0000   119 ;
                                0000   120 ; Multi-file summary line string, including cursor
                                0000   121 ; positioning and from/to times.
                                0000   122 ;
                                0000   123
                                0000   124 MF_SUMM1_STR::
                             2F' 0000   125                .BYTE   10$-5$
                       20 03 59 1B  0001   126 5$:           .BYTE   ESC,^A/Y/,3,32
53 20 45 4C 49 46 2D 49 54 4C 55 4D  0005   127                .ASCII  \MULTI-FILE SUMMARY\
                    59 52 41 4D 4D 55  0011
                       0E 06 59 1B  0017   128                .BYTE   ESC,^A/Y/,6,14
                    3A 65 64 6F 4E  001B   129                .ASCII  \Node:\
                       0E 07 59 1B  0020   130                .BYTE   ESC,^A/Y/,7,14
                    3A 6D 6F 72 46  0024   131                .ASCII  \From:\
                       0E 08 59 1B  0029   132                .BYTE   ESC,^A/Y/,8,14
                          3A 6F 54  002D   133                .ASCII  \To:\
                                0030   134 10$:
                                0030   135
                                0030   136 ;
                                0030   137 ; Multi-file summary statistic heading string (in smaller box).
                                0030   138 ;
                                0030   139
                                0030   140 MF_STATHEAD_STR::
                             21' 0030   141                .BYTE   10$-5$
                       0C 01 59 1B  0031   142 5$:           .BYTE   ESC,^A/Y/,1,12
                 2B 2D 2D 2D 2D 2D 2B  0035   143                .ASCII  \+-----+\
                       0C 02 59 1B  003C   144                .BYTE   ESC,^A/Y/,2,12
                 7C 20 45 56 41 20 7C  0040   145                .ASCII  \! AVE !\
                       0C 03 59 1B  0047   146                .BYTE   ESC,^A/Y/,3,12
                 2B 2D 2D 2D 2D 2D 2B  004B   147                .ASCII  \+-----+\
                                0052   148 10$:
                                0052   149
                                0052   150 ;
                                0052   151 ; Multi-file summary nodename heading control string
                                0052   152 ;
                                0052   153
                                0052   154 MFS_NODE_STR::
                             6E' 0052   155                .BYTE   10$-5$
                       1A 06 59 1B  0053   156 5$:           .BYTE   ESC,^A/Y/,6,26
28 3C 23 21 3E 21 43 41 21 3C 37 21  0057   157                .ASCII  \!7<!AC!>!#<(!UL)!>\
                    3E 21 29 4C 55 21  0063
                                0069   158 7$:
                       2D 06 59 1B  0069   159                .BYTE   ESC,^A/Y/,6,45
28 3C 23 21 3E 21 43 41 21 3C 37 21  006D   160                .ASCII  \!7<!AC!>!#<(!UL)!>\
                    3E 21 29 4C 55 21  0079
                       40 06 59 1B  007F   161                .BYTE   ESC,^A/Y/,6,64
28 3C 23 21 3E 21 43 41 21 3C 37 21  0083   162                .ASCII  \!7<!AC!>!#<(!UL)!>\
                    3E 21 29 4C 55 21  008F
                       53 06 59 1B  0095   163                .BYTE   ESC,^A/Y/,6,83
28 3C 23 21 3E 21 43 41 21 3C 37 21  0099   164                .ASCII  \!7<!AC!>!#<(!UL)!>\
                    3E 21 29 4C 55 21  00A5
                       66 06 59 1B  00AB   165                .BYTE   ESC,^A/Y/,6,102
28 3C 23 21 3E 21 43 41 21 3C 37 21  00AF   166                .ASCII  \!7<!AC!>!#<(!UL)!>\
```

```
                3E 21 29 4C 55 21   00BB
                                    00C1    167 10$:
                          00000016  00C1    168 MFS_NOD_SEGLEN == 7$-5$                     ; Length of one segment
                                    00C1    169
                                    00C1    170 ;
                                    00C1    171 ; Multi-file summary begin/end heading control string
                                    00C1    172 ;
                                    00C1    173
                                    00C1    174 MFS_TIME_STR::
                                37' 00C1    175             .BYTE   10$-5$
   3E 21 44 25 37 31 21 3C 39 31 21 00C2    176 5$:         .ASCII  \!19<!17%D!>\
                                    00CD    177 7$:
   3E 21 44 25 37 31 21 3C 39 31 21 00CD    178             .ASCII  \!19<!17%D!>\
   3E 21 44 25 37 31 21 3C 39 31 21 00D8    179             .ASCII  \!19<!17%D!>\
   3E 21 44 25 37 31 21 3C 39 31 21 00E3    180             .ASCII  \!19<!17%D!>\
   3E 21 44 25 37 31 21 3C 39 31 21 00EE    181             .ASCII  \!19<!17%D!>\
                                    00F9    182 10$:
                          0000000B  00F9    183 MFS_TIM_SEGLEN == 7$-5$                     ; Length of one segment
                                    00F9    184
                                    00F9    185 ;
                                    00F9    186 ; Multi-file summary statistics heading control string (1st)
                                    00F9    187 ;
                                    00F9    188
                                    00F9    189 MFS_STHEAD1_STR::
                                24' 00F9    190             .BYTE   10$-5$
            77 6F 52 20 2A 36 21    00FA    191 5$:         .ASCII  \!6* Row\
   20 2A 32 21 77 6F 52 20 2A 34 21 0101    192             .ASCII  \!4* Row!2* \
   20 2A 32 21 77 6F 52 20 2A 35 21 010C    193             .ASCII  \!5* Row!2* \
            77 6F 52 20 2A 35 21    0117    194             .ASCII  \!5* Row\
                                    011E    195 10$:
                                    011E    196
                                    011E    197 ;
                                    011E    198 ; Multi-file summary statistics heading control string (2nd)
                                    011E    199 ;
                                    011E    200
                                    011E    201 MFS_STHEAD2_STR::
                                28' 011E    202             .BYTE   10$-5$
            6D 75 53 20 2A 36 21    011F    203 5$:         .ASCII  \!6* Sum\
   65 67 61 72 65 76 41 20 2A 32 21 0126    204             .ASCII  \!2* Average\
   6D 75 6D 69 6E 69 4D 20 2A 33 21 0131    205             .ASCII  \!3* Minimum\
   6D 75 6D 69 78 61 4D 20 2A 33 21 013C    206             .ASCII  \!3* Maximum\
                                    0147    207 10$:
                                    0147    208
                                    0147    209
21 20 2A 30 31 21 0000014F'010E0000' 0147    210 DATA_STR:       .ASCID  \!10* !6UL.!2ZL\    ; Data substring for FAO ctrl str
               4C 5A 32 21 2E 4C 55 36  0155
2E 4C 55 36 21 20 00000165'010E0000' 015D    211 DATA_STR1:      .ASCID  \ !6UL.!2ZL\        ; Data substring for 1st column
                        4C 5A 32 21     016B
   20 2A 39 31 21 00000177'010E0000' 016F    212 BLANK_STR:      .ASCID  \!19* \             ; Blank substring for FAO ctrl str
   20 2A 30 31 21 00000184'010E0000' 017C    213 BLANK_STR1:     .ASCID  \!10* \             ; Blank substring for 1st column
2E 4C 55 37 21 20 00000191'010E0000' 0189    214 STATS_STR:      .ASCID  \ !7UL.!1ZL !6UL.!1ZL !6UL.!2ZL !6UL.!2ZL\
31 21 2E 4C 55 36 21 20 4C 5A 31 21  0197
4C 5A 32 21 2E 4C 55 36 21 20 4C 5A  01A3
      4C 5A 32 21 2E 4C 55 36 21 20  01AF
                                    01B9    215                                             ; Stats substring for FAO ctrl str
                                    01B9    216
                                    01B9    217
```

```
                                                H 16
00000000   218          .PSECT  MONDATA,QUAD,NOEXE
      0000   219
      0000   220 ;
      0000   221 ; OWN STORAGE (Writeable):
      0000   222 ;
      0000   223
      0000   224 MFSUMSTR::                                   ; FAO ctrl str segment to put one
   7F' 0000   225          .BYTE   10$-5$                     ; ... line of data
00000080  0001   226 5$:    .BLKB   127
      0080   227 10$:
      0080   228
```

```
                        .SBTTL  FILL_MFSUM_FAOSTK - Fill FAOSTK for M.F. Summary
                        .PSECT  $$MONCODE,NOWRT,EXE
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is called to fill the FAOSTK with values
                ;       to be presented to FAOL for output of the current page for the current
                ;       class. The address of the CDB for the current class is passed
                ;       as the first parameter to this routine.
                ;       Rest is TBS.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       CALLS #4,FILL_MFSUM_FAOSTK
                ;
                ; INPUTS:
                ;
                ;        4(AP) - address of a pointer to the CDB (Class Descriptor Block)
                ;                for the class to process.
                ;
                ;        8(AP) - address of byte containing starting column number.
                ;
                ;       12(AP) - address of byte containing number of columns.
                ;
                ;       16(AP) - address of byte containing statistics indicator.
                ;                If low-order bit = 0, no additional statistics are displayed.
                ;                If low-order bit = 1, additional statistics are displayed.
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       MRBPTR - pointer to MRB (Monitor Request Block)
                ;
                ;       MFSPTR - pointer to MFS (Multi-File Summary Block)
                ;
                ;       PERFTABLE - table describing each data item, indexed by
                ;                   item number ( * entry size)
                ;
                ;       FAOSTK - buffer into which to store values for later FAOL call.
                ;
                ; OUTPUTS:
                ;
                ;       None.
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       FAOSTK buffer is filled with arguments for an $FAOL call
                ;       to display one page of the multi-file summary report.
                ;
                ;       MFS$L_LWORDS set to number of longwords on FAOSTK for
                ;       each element.
                ;
                ; ROUTINE VALUE:
                ;
                ;       R0 = SS$_NORMAL
                ;
                ; SIDE EFFECTS:
```

```
0000    287 ;
0000    288 ;       None
0000    289 ;
0000    290 ; REGISTER USAGE:
0000    291 ;
0000    292 ;       R0 = address of item number
0000    293 ;       R1 = scratch, also input to CVTSTK subroutine
0000    294 ;       R2 = element index
0000    295 ;       R3 = current column number
0000    296 ;       R4 = address of data for current column in summary buffer
0000    297 ;       R5 = pointer to current longword in FAOSTK
0000    298 ;       R6 = address of CDB for class to display
0000    299 ;       R7 = address of Multi-File Summary Block (MFS)
0000    300 ;       R8 = address of temporary local data area
0000    301 ;       R9 = Number of elements to examine for this class
0000    302 ;       R10 = address of summary buffer for this class
0000    303 ;       R11 = scratch, in CVTSTK subroutine
0000    304 ;
0000    305 ;--
0000    306
```

```
                     0000   308
                0FFC 0000   309 .ENTRY  FILL_MFSUM_FAOSTK, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     0002   310
      56   04 BC   D0  0002   311         MOVL    @4(AP),R6               ; Load CDB pointer
                     0006   312
                     0006   313         ALLOC   TM2$K_SIZE,R0,R8        ; Allocate local temp storage
                     0013   314
50   00000000'EF    D0  0013   315         MOVL    MFSPTR,R0               ; Get MFS pointer
          34 A0     D4  001A   316         CLRL    MFS$L_LWORDS(R0)        ; Clear count of longwords on FAOSTK
   18 A8  30 A0     D0  001D   317         MOVL    MFS$L_ELEMS(R0),TM2$L_ELEMS(R8) ; Get elements to display
             03     12  0022   318         BNEQ    10$                     ; Br if have some
           0129     31  0024   319         BRW     FMF_RET                 ; Else go exit if none
                    0027   320 10$:
                    0027   321
      68   08 BC   9A  0027   322         MOVZBL  @8(AP),TM2$L_START_COL(R8) ; Get starting column number
             68     D7  002B   323         DECL    TM2$L_START_COL(R8)     ; ... and make it zero-origin
   04 A8   0C BC   9A  002D   324         MOVZBL  @12(AP),TM2$L_COLS(R8)  ; Get number of columns to display
                    0032   325
                    0032   326 ;
                    0032   327 ; Compute size of a column's worth of data in bytes
                    0032   328 ; (to be used later).
                    0032   329 ;
                    0032   330
      50   18 A8   D0  0032   331         MOVL    TM2$L_ELEMS(R8),R0      ; Get # of lwords for a column (hetero)
   07 4B A6    05   E1  0036   332         BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),15$ ; Br if heterogeneous class
50   00000000'8F    D0  003B   333         MOVL    #MAXELTS_MFS,R0         ; Get # of lwords for a column (homog)
                    0042   334 15$:
 08 A8   50   02   C1  0042   335         ADDL3   #2,R0,TM2$L_COL_SIZE(R8) ; Include secs & collection counters
      08 A8   04   C4  0047   336         MULL2   #4,TM2$L_COL_SIZE(R8)   ; ... translate to bytes
                    004B   337
      5A   0C A6   D0  004B   338         MOVL    CDB$A_SUMBUF(R6),R10    ; Load address of summary buffer
```

```
                                                L 16
                     004F   340
                     004F   341 ;
                     004F   342 ; For homogeneous classes, get summary buffer address;
                     004F   343 ; then determine item type
                     004F   344 ;
                     004F   345
                     004F   346
 31 4B A6    05   E1 004F   347          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),20$ ; Br if not homog class
    51    32 A6   D0 0054   348          MOVL    CDB$A_CDX(R6),R1          ; Get CDX address
    50    07 A1   9A 0058   349          MOVZBL  CDX$B_IDISCONSEC(R1),R0 ; Get no. of curr display item
             50   D7 005C   350          DECL    R0                        ; Decrement to make it zero-origin
    53    06 A1   9A 005E   351          MOVZBL  CDX$B_IDISCT(R1),R3       ; Get number of items being displayed
 53    08 A6    53   C7 0062   352          DIVL3   R3,CDB$L_SUMBUF(R6),R3  ; Compute size of sumbuf for one item
                     0067   353                                            ; Note -- cannot have zero items
       53    50   C4 0067   354          MULL2   R0,R3                     ; Compute distance to desired sumbuf
    5A    6A43   9E 006A   355          MOVAB   (R10)[R3],R10             ; Get summ buff ptr for homog class
                     006E   356
                     006E   357 ;
                     006E   358 ; Now determine item type.
                     006E   359 ;
                     006E   360
    53    08 A1   9A 006E   361          MOVZBL  CDX$B_IDISINDEX(R1),R3  ; Get item index for this disp event
 50    1C B643   9A 0072   362          MOVZBL  @CDB$A_ITMSTR(R6)[R3],R0 ; Load IDB item number
       50    11   C4 0077   363          MULL2   #IDB$K_ILENGTH,R0         ; Compute index into IDB table
50    0000'CF40   9E 007A   364          MOVAB   W^PERFTABLE[R0],R0        ; Address of IDB for this item
 1C A8    0A A0   B0 0080   365          MOVW    IDB$W_TYPE(R0), -
                     0085   366                  TM2$W_ITEM_TYPE(R8)       ; Save item type for use below
                     0085   367
                     0085   368 20$:
          0194   30 0085   369          BSBW    ESTAB_FAOSTR              ; Establish FAO ctrl substring
                     0088   370                                            ; Note -- destroys all regs exc.
                     0088   371                                            ; ... R6,R8,R10
                     0088   372
```

```
                                0088   374 ;
                                0088   375 ; Loop once for each element in this class, processing the summary buffer.
                                0088   376 ; For each column, convert the "sum" number to an "average" by dividing
                                0088   377 ; by seconds or collections. Place the result in FAOSTK, as longword
                                0088   378 ; arguments for a subsequent call to $FAOL to display a single page of
                                0088   379 ; the multi-file summary report.
                                0088   380 ;
                                0088   381
55   00000000'EF    DE          0088   382          MOVAL   FAOSTK,R5                ; Load addr of buffer for FAO parms
57   00000000'EF    D0          008F   383          MOVL    MFSPTR,R7                ; Get MFS pointer
                                0096   384
 59   18 A8   01    C3          0096   385          SUBL3   #1,TM2$L_ELEMS(R8),R9    ; Get number of elts to examine (loop limit)
   09 4B A6   08    E1          009B   386          BBC     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),30$ ; Br if not SYSTEM class
50   00000000'EF    DE          00A0   387          MOVAL   ITMSTR_SYS_ALL,R0        ; SYS/ALL itemstring is what we want
                04    11          00A7   388          BRB     35$                      ; skip past use of CDB itemstring
                                00A9   389 30$:
      50   1C A6    D0          00A9   390          MOVL    CDB$A_ITMSTR(R6),R0      ; Address of item-number string
                                00AD   391 35$:
                52    D4          00AD   392          CLRL    R2                       ; Clear element index register
                                00AF   393
                                00AF   394 FMF_ELEM:
   11 4B A6    05    E0          00AF   395          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if a homogeneous class
         51    80    9A          00B4   396          MOVZBL  (R0)+,R1                 ; Get next item number
         51    11    C4          00B7   397          MULL2   #IDB$K_ILENGTH,R1        ; Compute index into IDB table
5B   0000'CF41    9E          00BA   398          MOVAB   W^PERFTABLE[R1],R11      ; Address of IDB for this item
 1C A8   0A AB    B0          00C0   399          MOVW    IDB$W_TYPE(R11), -
                                00C5   400                  TM2$W_ITEM_TYPE(R8)      ; Save item type for use below
                                00C5   401 10$:
      04 A8    D5          00C5   402          TSTL    TM2$L_COLS(R8)           ; Any columns to display ?
         03    12          00C8   403          BNEQ    20$                      ; Br if some
       0067    31          00CA   404          BRW     70$                      ; Else skip loop for this element
                                00CD   405 20$:
                                00CD   406
   53   68   01    C3          00CD   407          SUBL3   #1,TM2$L_START_COL(R8),R3 ; Get starting col. no. (0-origin)
                                00D1   408                                           ; ... and make it 1 less than that.
                                00D1   409                                           ; ... will regain 1 within loop
51   53   08 A8    C5          00D1   410          MULL3   TM2$L_COL_SIZE(R8),R3,R1 ; Compute addr of data
  54   5A   51    C1          00D6   411          ADDL3   R1,R10,R4                ; ... for starting column
        0C A8    D4          00DA   412          CLRL    TM2$L_COLS_USED(R8)      ; Init counter of columns used
                                00DD   413 30$:
         53    D6          00DD   414          INCL    R3                       ; Point to next column number
      54   08 A8    C0          00DF   415          ADDL2   TM2$L_COL_SIZE(R8),R4    ; ... and data for next column
                                00E3   416
51   00000000'EF    D0          00E3   417          MOVL    CSBVEC_PTR,R1            ; Get ptr to CSB vector
       51   6143    D0          00EA   418          MOVL    (R1)[R3],R1              ; ... and CSB for current column
   EA 20 A1   00    E0          00EE   419          BBS     #CSB$V_IGNORE,CSB$B_FLAGS(R1),30$ ; Ignore column if instructed
51   54   08 A8    C1          00F3   420          ADDL3   TM2$L_COL_SIZE(R8),R4,R1 ; Point to first byte after this column
  10 A8   F8 A1    D0          00F8   421          MOVL    -8(R1),TM2$L_SECONDS(R8) ; Get seconds
  14 A8   FC A1    D0          00FD   422          MOVL    -4(R1),TM2$L_COLLS(R8)   ; Get collection count
         2A    13          0102   423          BEQL    60$                      ; Br if zero collections
   0A 4B A6    05    E0          0104   424          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),35$ ; Br if a homogeneous class
         00    E1          0109   425          BBC     #IDB$V_PCNT,-            ; Branch if this is not
      05 10 AB                010B   426                  IDB$B_FLAGS(R11),35$     ; a percent type computed item
       0047    30          010E   427          BSBW    COMPUTED_ITEM_AVE        ; get average for computed item
         16    11          0111   428          BRB     50$                      ; and continue (R1 now contains average)
                                0113   429 35$:
      51   6442    D0          0113   430          MOVL    (R4)[R2],R1              ; Get floating sum of interest
```

```
0000'8F   1C A8   B1  0117   431          CMPW    TM2$W_ITEM_TYPE(R8),#COUNT_TYPE ; This item a count?
          06   12  011D   432          BNEQ    40$                              ; No -- assume level type
       51 10 A8   46  011F   433          DIVF    TM2$L_SECONDS(R8),R1             ; Yes -- get floating avg rate/second
          04   11  0123   434          BRB     50$                              ; ... and continue
```

```
                        0125   436 40$:
          51    14 A8   46  0125   437          DIVF    TM2$L_COLLS(R8),R1      ; Compute floating avg size/collection
                        0129   438 50$:
                  51    10  0129   439          BSB     CVTSTK                  ; Convert and stack this number (in R1)
                0080    30  012B   440          BSBW    UPD_STATS               ; Update MIN, MAX and TOT
                        012E   441 60$:
    A9 0C A8    04 A8   F2  012E   442          AOBLSS  TM2$L_COLS(R8), -
                        0134   443                  TM2$L_COLS_USED(R8),30$ ; ... and go back to get next column
                        0134   444 70$:
       03 10 BC   00    E1  0134   445          BBC     #0,@16(AP),80$          ; Skip stacking stats if not requested
                00A2    30  0139   446          BSBW    STACK_STATS             ; Convert & stack all 4 stats
                        013C   447 80$:
       09 4B A6   05    E0  013C   448          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),85$ ; Br if a homogeneous class
                00      E1  0141   449          BBC     #IDB$V_PCNT,-           ; Branch if this is not
             04 10 AB           0143   450                  IDB$B_FLAGS(R11),85$    ; a percent type computed item
                50      D6  0146   451          INCL    R0                      ; increment index into item table
                52      D6  0148   452          INCL    R2                      ; increment index into data buffer
                        014A   453 85$:
  FF5F 52   01    59    F1  014A   454          ACBL    R9,#1,R2,FMF_ELEM       ; Loop once for each element
                        0150   455
                        0150   456 FMF_RET:
   50   00000000'8F     D0  015C   457          MOVL    #SS$_NORMAL,R0          ; Indicate success
                04      0157   458          RET                             ; ... and return
                        0158   459
                        0158   460 ;
                        0158   461 ;
                        0158   462 ; COMPUTED_ITEM_AVE - Do average computation for computed item
                        0158   463 ;       R2 = element index
                        0158   464 ;       R4 = address of data for current column in summary buffer
                        0158   465 ;       R1 (output) - calculated average
                        0158   466 ;       R10-R11 scratch (saved on stack)
                        0158   467 ;
                        0158   468
                        0158   469 COMPUTED_ITEM_AVE:
                        0158   470
                    5A  DD  0158   471          PUSHL   R10                     ; save R10
                    5B  DD  015A   472          PUSHL   R11                     ; save R11
                    51  D4  015C   473          CLRL    R1                      ; assume zero average
  5A   000043C8 8F    04 A442  45  015E   474          MULF3   4(R4)[R2],-             ; sum for item1 *100 into R10
                        0168   475                  #100, R10               ;
                5B  08 A442  50  0168   476          MOVF    8(R4)[R2],R11           ; item2 into R11
                    5B  53  016D   477          TSTF    R11                     ; is R11 0?
                    04  13  016F   478          BEQL    10$                     ; skip divide if so
                51  5A  5B  47  0171   479          DIVF3   R11,R10,R1              ; result of divide into R1
                        0175   480 10$:
                    5B 8ED0 0175   481          POPL    R11                     ; restore registers
                    5A 8ED0 0178   482          POPL    R10
                        05  017B   483          RSB                             ; exit
                        017C   484 ;
                        017C   485 ; CVTSTK - Convert floating longword input value to two longword
                        017C   486 ;                  integer values representing the whole and fractional
                        017C   487 ;                  parts. Also, place both integer longwords into the
                        017C   488 ;                  next available slots in the FAOSTK.
                        017C   489 ;
                        017C   490 ; CVTSTK10 is an alternate entry point which stacks a fractional
                        017C   491 ;                  part which is in tenths instead of the usual hundredths.
                        017C   492 ;
```

```
                                017C   493 ;
                                017C   494 ;               R1  = input floating value to convert and stack
                                017C   495 ;               R5  = address of next available longword in FAOSTK
                                017C   496 ;               R11 = scratch (saved on stack)
                                017C   497 ;
                                017C   498
                                017C   499 CVTSTK:
                    5B    DD    017C   500          PUSHL   R11                     ; save R11 so we can use it as scratch
                 85     51    4A 017E   501          CVTFL   R1,(R5)+                ; Stack whole part for fao
              5B     FC A5    4E 0181   502          CVTLF   -4(R5),R11              ; Get back truncated part
           5B    51    5B    43 0185   503          SUBF3   R11,R1,R11              ; Compute fraction to two ...
        5B    000043C8 8F    44 0189   504          MULF2   #100,R11                ; ... digits for tabular display
                 85    5B    4A 0190   505          CVTFL   R11,(R5)+               ; Stack fraction for fao
                       5B 8ED0  0193   506          POPL    R11                     ; Restore R11
                          05    0196   507          RSB
                                0197   508
                                0197   509 CVTSTK10:
                    5B    DD    0197   510          PUSHL   R11                     ; save R11 so we can use it as scratch
                 85     51    4A 0199   511          CVTFL   R1,(R5)+                ; Stack whole part for fao
              5B     FC A5    4E 019C   512          CVTLF   -4(R5),R11              ; Get back truncated part
           5B    51    5B    43 01A0   513          SUBF3   R11,R1,R11              ; Compute fraction to one ...
                 5B    22    44 01A4   514          MULF2   #10,R11                 ; ... digit for tabular display
                 85    5B    4A 01A7   515          CVTFL   R11,(R5)+               ; Stack fraction for fao
                       5B 8ED0  01AA   516          POPL    R11                     ; Restore R11
                          05    01AD   517          RSB
```

```
                        01AE    519 ;
                        01AE    520 ; UPD_STATS - Update the MIN, MAX, and TOT longwords for this element.
                        01AE    521 ;
                        01AE    522 ;               R1  = input floating value to update 3 stats with.
                        01AE    523 ;               R2  = current element number (0-origin)
                        01AE    524 ;               R7  = MFS pointer
                        01AE    525 ;               R8  = scratch (saved on stack)
                        01AE    526 ;               R9  = scratch (saved on stack)
                        01AE    527 ;
                        01AE    528
                        01AE    529 UPD_STATS:
                        01AE    530
      0300 8F   BB      01AE    531             PUSHR   #^M<R8,R9>                      ; Save regs
59  20 A7   03   C7      01B2    532             DIVL3   #3,MFS$L_STATSBUF(R7),R9 ; Get length of one buffer
    58   24 A7   D0      01B7    533             MOVL    MFS$A_STATSBUF(R7),R8   ; Get ptr to first buff (TOT)
                        01BB    534
     6842   51   40      01BB    535             ADDF2   R1,(R8)[R2]             ; Update TOT for this element
                        01BF    536
       58   59   C0      01BF    537             ADDL2   R9,R8                   ; Compute addr of MIN buffer
     6842   51   51      01C2    538             CMPF    R1,(R8)[R2]             ; Check minimum
            04   18      01C6    539             BGEQ    10$                     ; Branch if not less
     6842   51   50      01C8    540             MOVF    R1,(R8)[R2]             ; Else insert new minimum
                        01CC    541 10$:
       58   59   C0      01CC    542             ADDL2   R9,R8                   ; Compute addr of MAX buffer
     6842   51   51      01CF    543             CMPF    R1,(R8)[R2]             ; Check maximum
            04   15      01D3    544             BLEQ    20$                     ; Branch if not more
     6842   51   50      01D5    545             MOVF    R1,(R8)[R2]             ; Else insert new maximum
                        01D9    546 20$:
      0300 8F   BA      01D9    547             POPR    #^M<R8,R9>              ; Restore regs
                05      01DD    548             RSB                             ; Return
                        01DE    549
```

```
                     01DE   551 ;
                     01DE   552 ; STACK_STATS - Convert and stack each of the four statistics
                     01DE   553 ;               (TOT, AVE, MIN, MAX) for the current element.
                     01DE   554 ;
                     01DE   555 ;               R1 = input to CVTSTK (altered by this routine)
                     01DE   556 ;               R2  = current element number (0-origin)
                     01DE   557 ;               R5  = address of next available longword in FAOSTK
                     01DE   558 ;               R7  = MFS pointer
                     01DE   559 ;               R8 - R10 = scratch (saved on stack)
                     01DE   560 ;
                     01DE   561
                     01DE   562 STACK_STATS:
                     01DE   563
      0700 8F    BB  01DE   564         PUSHR   #^M<R8,R9,R10>          ; Save regs
 5A     3C A7    9A  01E2   565         MOVZBL  MFS$B_DATA_COLS(R7),R10 ; Get # of cols with actual data
59  20 A7  03    C7  01E6   566         DIVL3   #3,MFS$L_STATSBUF(R7),R9 ; Get length of one buffer
58    24 A7    D0  01EB   567         MOVL    MFS$A_STATSBUF(R7),R8   ; Get ptr to 1st buff (TOT)
                     01EF   568
 51    6842    D0  01EF   569         MOVL    (R8)[R2],R1             ; Get TOT value for this element
             A2    10  01F3   570         BSB     CVTSTK10                ; Convert and stack it (in tenths)
                     01F5   571
    5A    5A    4E  01F5   572         CVTLF   R10,R10                 ; Get flt. # of cols with actual data
             04    12  01F8   573         BNEQ    10$                     ; Br if have some
             51    D4  01FA   574         CLRL    R1                      ; No data columns -- just stack 0
             03    11  01FC   575         BRB     20$                     ; Go do it
                     01FE   576 10$:
    51    5A    46  01FE   577         DIVF2   R10,R1                  ; Compute AVE value for this element
                     0201   578 20$:
             94    10  0201   579         BSB     CVTSTK10                ; Convert and stack AVE value (tenths)
                     0203   580
   58    59    C0  0203   581         ADDL2   R9,R8                   ; Compute addr of MIN buffer
51    6842    D0  0206   582         MOVL    (R8)[R2],R1             ; Get MIN value for this element
      FF6F    30  020A   583         BSBW    CVTSTK                  ; Convert and stack it
                     020D   584
   58    59    C0  020D   585         ADDL2   R9,R8                   ; Compute addr of MAX buffer
51    6842    D0  0210   586         MOVL    (R8)[R2],R1             ; Get MAX value for this element
      FF65    30  0214   587         BSBW    CVTSTK                  ; Convert and stack it
                     0217   588
  0700 8F    BA  0217   589         POPR    #^M<R8,R9,R10>          ; Restore regs
             05  021B   590         RSB                             ; Return
```

```
                                021C   592 ;
                                021C   593 ; ESTAB_FAOSTR - Establish the MFSUMSTR counted string, which is the
                                021C   594 ;                substring to be used by the TEMPLATE routine to build
                                021C   595 ;                an FAO control string to display one line of data.
                                021C   596 ;                Also, compute MFS$L_LWORDS for later use by DISPLAY_HOMOG
                                021C   597 ;                routine (for homog classes). It is the number of longwords
                                021C   598 ;                which will be put on the FAOSTK by FILL_MFSUM_FAOSTK for
                                021C   599 ;                each element.
                                021C   600 ;
                                021C   601 ;       Upon input,
                                021C   602 ;
                                021C   603 ;               R0 - R5,R7,R9,R11 = scratch, destroyed by this subroutine
                                021C   604 ;               R6  = CDB pointer
                                021C   605 ;               R8  = temporary data area pointer
                                021C   606 ;               R10 = SUMM BUFF pointer
                                021C   607 ;
                                021C   608
                                021C   609 ESTAB_FAOSTR:
                                021C   610
53   00000001'EF   9E           021C   611         MOVAB   MFSUMSTR+1,R3           ; Get addr of MFSUMSTR string
           04 A8   D5           0223   612         TSTL    TM2$L_COLS(R8)          ; Any columns to display ?
              7D   13           0226   613         BEQL    60$                     ; Br if none
   57   68   01   C3           0228   614         SUBL3   #1,TM2$L_START_COL(R8),R7 ; Get starting col. no. (0-origin)
                                022C   615                                         ; ... and make it 1 less than that.
                                022C   616                                         ; ... will regain 1 within loop
51 57   08 A8   C5           022C   617         MULL3   TM2$L_COL_SIZE(R8),R7,R1 ; Compute addr of data
   59   5A   51   C1           0231   618         ADDL3   R1,R10,R9               ; ... for starting column
              5B   D4           0235   619         CLRL    R11                     ; Init counter of columns used
                                0237   620 10$:
              57   D6           0237   621         INCL    R7                      ; Get column number
           59   08 A8   C0      0239   622         ADDL2   TM2$L_COL_SIZE(R8),R9   ; Point to data for this column
                                023D   623
      51   00000000'FF47   D0   023D   624         MOVL    @CSBVEC_PTR[R7],R1      ; ... and CSB for this column
        ED 20 A1   00   E0      0245   625         BBS     #CSB$V_IGNORE,CSB$B_FLAGS(R1),10$ ; Ignore column if instructed
   51   59   08 A8   C1      024A   626         ADDL3   TM2$L_COL_SIZE(R8),R9,R1 ; Point to first byte after this column
              FC A1   D5      024F   627         TSTL    -4(R1)                  ; Any collections?
                 2E   13      0252   628         BEQL    30$                     ; Br if no
   50   00000000'EF   D0      0254   629         MOVL    MFSPTR,R0               ; Get MFS pointer
              3C A0   96      025B   630         INCB    MFS$B_DATA_COLS(R0)     ; Count this column with data in it
           34 A0   02   C0      025E   631         ADDL2   #2,MFS$L_LWORDS(R0)     ; Count longwords to be added to FAOSTK
              5B   D5      0262   632         TSTL    R11                     ; First column ?
                 0E   13      0264   633         BEQL    20$                     ; Yes -- br to move a special string
63   0000014F'EF   00000147'EF   28   0266   634         MOVC3   DATA_STR,DATA_STR+8,(R3) ; Move data substring into ctrl str
                 2C   11      0272   635         BRB     50$                     ; ... and go loop
                                0274   636 20$:
63   00000165'EF   0000015D'EF   28   0274   637         MOVC3   DATA_STR1,DATA_STR1+8,(R3) ; Move speciai data substring
                 1E   11      0280   638         BRB     50$                     ; ... and go loop
                                0282   639 30$:
              5B   D5      0282   640         TSTL    R11                     ; First Column ?
                 0E   13      0284   641         BEQL    40$                     ; Yes -- br to move a special string
63   00000177'EF   0000016F'EF   28   0286   642         MOVC3   BLANK_STR,BLANK_STR+8,(R3) ; Move blank substring into ctrl str
                 0C   11      0292   643         BRB     50$                     ; ... and go loop
                                0294   644 40$:
63   00000184'EF   0000017C'EF   28   0294   645         MOVC3   BLANK_STR1,BLANK_STR1+8,(R3) ; Move special blank substring
                                02A0   646 50$:
           92 5B   04 A8   F2   02A0   647         AOBLSS  TM2$L_COLS(R8),R11,10$  ; ... and go back to get next column
```

```
                                02A5   649
                                02A5   650 60$:
                     17 10 BC   00  E1  02A5   651          BBC     #0,@16(AP),70$          ; Skip if stats not requested
                                02AA   652
                50   00000000'EF    D0  02AA   653          MOVL    MFSPTR,R0               ; Get MFS pointer
                      34 A0    08    C0  02B1   654          ADDL2   #8,MFS$L_LWORDS(R0)     ; Count longwords to be added to FAOSTK
63   00000191'EF   00000189'EF    28  02B5   655          MOVC3   STATS_STR,STATS_STR+8,(R3) ; Move stats substring into ctrl str
                                02C1   656 70$:
                50   00000001'EF    9E  02C1   657          MOVAB   MFSUMSTR+1,R0           ; Get addr of MFSUMSTR string
                     50   53    50    C3  02C8   658          SUBL3   R0,R3,R0                ; Compute actual length of string
                00000000'EF   50    90  02CC   659          MOVB    R0,MFSUMSTR             ; ... and store in cstring
                                05  02D3   660          RSB                             ; Return
```

```
                                                I  1
02D4    662          .SBTTL  CAPTURE_SUMS - Move SUM buffers to Summary Buffer
02D4    663
02D4    664 ;++
02D4    665 ;
02D4    666 ; FUNCTIONAL DESCRIPTION:
02D4    667 ;
02D4    668 ;       This routine updates the multi-file summary buffer with
02D4    669 ;       the contribution of the current class from the current
02D4    670 ;       input file. Each sum in the SUM buffer is converted to
02D4    671 ;       floating-point, and either added or moved to the m.f.
02D4    672 ;       summary buffer. Also, the number of seconds for the current
02D4    673 ;       class and file, as well as the number of collections,
02D4    674 ;       are also converted and moved (or added).  For the SYSTEM class,
02D4    675 ;       a SUM buff for SYS/ALL is constructed "on the fly".
02D4    676 ;
02D4    677 ;
02D4    678 ; CALLING SEQUENCE:
02D4    679 ;
02D4    680 ;       CALLS   #2,CAPTURE_SUMS
02D4    681 ;
02D4    682 ; INPUTS:
02D4    683 ;
02D4    684 ;        4(AP) - address of a pointer to the CDB (Class Descriptor Block)
02D4    685 ;                for the class to process.
02D4    686 ;
02D4    687 ;        8(AP) - address of byte containing column number (1-origin).
02D4    688 ;
02D4    689 ; IMPLICIT INPUTS:
02D4    690 ;
02D4    691 ;       MRBPTR - pointer to MRB (Monitor Request Block)
02D4    692 ;
02D4    693 ;       MCAPTR - pointer to MCA (Monitor Communication Area)
02D4    694 ;
02D4    695 ; OUTPUTS:
02D4    696 ;
02D4    697 ;       None
02D4    698 ;
02D4    699 ; IMPLICIT OUTPUTS:
02D4    700 ;
02D4    701 ;       The multi-file summary buffer is updated to reflect the inclusion
02D4    702 ;       of SUM data for the current class and input file.
02D4    703 ;
02D4    704 ; ROUTINE VALUE:
02D4    705 ;
02D4    706 ;       R0 = SS$_NORMAL, or failing MONITOR status
02D4    707 ;
02D4    708 ; SIDE EFFECTS:
02D4    709 ;
02D4    710 ;       None
02D4    711 ;
02D4    712 ; REGISTER USAGE:
02D4    713 ;
02D4    714 ;       R0 = scratch
02D4    715 ;       R1 = scratch, also floating seconds
02D4    716 ;       R2 = floating collection count
02D4    717 ;       R3 = address of current element in summary buffer
02D4    718 ;       R4 = scratch, no. of elements in this class
```

```
                                                     J 1
                                02D4   719 ;       R5 = scratch, also address of SUM buffer for this input file
                                02D4   720 ;       R6 = address of CDB for this class
                                02D4   721 ;       R7 = address of MRB
                                02D4   722 ;
                                02D4   723 ;--
                                02D4   724
                        07FC    02D4   725 .ENTRY  CAPTURE_SUMS, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
                                02D6   726
                    56  04 BC   D0    02D6   727         MOVL    @4(AP),R6               ; Get ptr to CDB
                57  00000000'EF   D0    02DA   728         MOVL    MRBPTR,R7               ; Get ptr to MRB
                                02E1   729
                                02E1   730
                                02E1   731 ;
                                02E1   732 ; Compute the number of floating seconds and the number of
                                02E1   733 ; floating collections covered by the current input file,
                                02E1   734 ; and save for use below.
                                02E1   735 ;
                                02E1   736
                    51  0C A7   D0    02E1   737         MOVL    MRB$Q_ENDING+4(R7),R1 ; Get high order bits of time
                 50  08 A7  67   C3    02E5   738         SUBL3   MRB$Q_BEGINNING(R7),MRB$Q_ENDING(R7),R0 ; Compute elapsed time since
                    51  04 A7   D9    02EA   739         SBWC    MRB$Q_BEGINNING+4(R7),R1 ; Get high order difference
51  50  50  00002710 8F   7B    02EE   740         EDIV    #10000,R0,R0,R1         ; Turn time into milliseconds
                50  50   4E    02F7   741         CVTLF   R0,R0                   ; Floating milliseconds
    51  50  0000457A 8F   47    02FA   742         DIVF3   #1000,R0,R1             ; Save floating seconds for later use
                                0302   743
                50  00000000'EF   D0    0302   744         MOVL    MCAPTR,R0               ; Get ptr to MCA
                 52  0C A0  01   C3    0309   745         SUBL3   #1,MCA$L_COLLCNT(R0),R2 ; No. of collections, don't count 1st
                    52  52   4E    030E   746         CVTLF   R2,R2                   ; Convert to floating for later use
```

```
                                0311   748 ;
                                0311   749 ; Compute the starting address of
                                0311   750 ; the portion of the output m.f. summary buffer for this column
                                0311   751 ; (a column represents a column of data on the report page --
                                0311   752 ; it can be the data for a single file, or, if BY_NODE was
                                0311   753 ; requested, for a single node).
                                0311   754 ;
                                0311   755
        53   0C A6    D0         0311   756          MOVL    CDB$A_SUMBUF(R6),R3      ; Get beginning of m.f. summary buffer
        54   08 BC    9A         0315   757          MOVZBL  @8(AP),R4                ; Get column number
             54       D7         0319   758          DECL    R4                       ; Make it 0-origin
             2C       13         031B   759          BEQL    20$                      ; Skip calcs if column 1
55   00000008'8F      D0         031D   760          MOVL    #<4*<MAXELTS_MFS+2>>,R5 ; Compute no. of bytes for this column
   19 4B A6           05         0324   761          BBS     #CDB$V_HOMOG,CDB$L_FLAGS(R6),10$ ; Br if a homog class
   0C 4B A6           08         0329   762          BBC     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),7$         ; Br if not SYSTEM class
55   00000002'8F      D0         032E   763          MOVL    #<ECOUNT_SYS_ALL+2>,R5  ; SYSTEM class item count
        55            04         C4   0335   764          MULL2   #4,R5
             08       11         0338   765          BRB     10$
                                033A   766 7$:
 55   18 A6           02         C1   033A   767          ADDL3   #2,CDB$L_ECOUNT(R6),R5  ; Re-compute for hetero
        55            04         C4   033F   768          MULL2   #4,R5                   ; ....
                                0342   769 10$:
        55            54         C4   0342   770          MULL2   R4,R5                   ; No. of bytes to skip past
        53          6345         9E   0345   771          MOVAB   (R3)[R5],R3             ; Compute start address of column
                                0349   772 20$:
   12 4B A6           05         E1   0349   773          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),30$ ; Br if a heterogeneous class
                                034E   774
                                034E   775 ;
                                034E   776 ; For homogeneous classes, call special routine
                                034E   777 ; to fill summary buffer.
                                034E   778 ;
                                034E   779
             52       DD         034E   780          PUSHL   R2                       ; Stack floating collections
             51       DD         0350   781          PUSHL   R1                       ; Stack floating seconds
             53       DD         0352   782          PUSHL   R3                       ; Stack start address of column
             56       DD         0354   783          PUSHL   R6                       ; Stack CDB address
000003FD'EF   04       FB         0356   784          CALLS   #4,FILL_HOM_SUMMBUFF    ; Fill summary buffer for homog class
          009C         31         035D   785          BRW     CS_RET                  ; Go return with status
                                0360   786
                                0360   787 ;
                                0360   788 ; Fill summary buffer for heterogeneous class
                                0360   789 ;
                                0360   790
                                0360   791 30$:
   62 4B A6           08         E1   0360   792          BBC     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),70$       ; Br if not SYSTEM class
58   00000000'8F      D0         0365   793          MOVL    #<CDB$K_SIZE*MODES_CLSNO>,R8 ; Compute offset to MODES CDB
58   00000000'EF48    9E         036C   794          MOVAB   CDBHEAD[R8],R8          ; ... get its CDB address
        59    2E A8    D0         0374   795          MOVL    CDB$A_BUFFERS(R8),R9    ; ... and MBP ptr
        59    14 A9    D0         0378   796          MOVL    MBP$A_SUM(R9),R9        ; ...point to SUM buff for MODE
        55    2E A6    D0         037C   797          MOVL    CDB$A_BUFFERS(R6),R5    ; Get addr of SYSTEM class buffers
   55   14 A5           08         C1   0380   798          ADDL3   #8,MBP$A_SUM(R5),R5     ; Get SYSTEM SUM buff ptr (skips 2 itemS)
                                0385   799 ;
                                0385   800 ; Now loop once for each element, moving or adding data from
                                0385   801 ; the MODES and SYSTEM SUM buffer to the m.f. summary buffer.
                                0385   802 ;
   1E 43 A7           0C         E0   0385   803          BBS     #MRB$V_BY_NODE,MRB$W_FLAGS(R7),50$ ; Go do a sum if by node
                                038A   804 40$:
```

```
                54   07   D0  038A   805          MOVL    #7,R4                           ; Get no. of elements for MODES CLASS
                                038D   806 45$:
                83   89   4E  038D   807          CVTLF   (R9)+,(R3)+                     ; Fill summary buffer for this element with
                  FA 54   F5  0390   808          SOBGTR  R4,45$                          ; Loop to pick up all MODES items
                                0393   809
            54  FFFFFFF9'8F  D0  0393   810          MOVL    #<ECOUNT_SYS_ALL-7>,R4  ; no. of items to pick up from SYSTEM class
                                039A   811 47$:
                83   85   4E  039A   812          CVTLF   (R5)+,(R3)+                     ; Fill summary buffer for this element with
                  FA 54   F5  039D   813          SOBGTR  R4,47$                          ; Loop to pick up all SYSTEM items
                                03A0   814
                83   51   D0  03A0   815          MOVL    R1,(R3)+                        ; Also move in floating seconds
                63   52   D0  03A3   816          MOVL    R2,(R3)                         ; ... and collection count
                     4D   11  03A6   817          BRB     CS_NORMAL                       ; ... and go return
                                03A8   818
                                03A8   819 50$:                                         ; BY_NODE logic for SYSTEM class
                54   07   D0  03A8   820          MOVL    #7,R4                           ; Get no. of elements for this class
                                03AB   821 55$:
                50   89   4E  03AB   822          CVTLF   (R9)+,R0                        ; Augment summary buff for this element
                83   50   40  03AE   823          ADDF2   R0,(R3)+                        ; ... with this file's contribution from MOD
                  F7 54   F5  03B1   824          SOBGTR  R4,55$                          ; Loop to do entire column
                                03B4   825
            54  FFFFFFF9'8F  D0  03B4   826          MOVL    #<ECOUNT_SYS_ALL-7>,R4  ; no. of items to pick up from SYSTEM class
                                03BB   827 57$:
                50   85   4E  03BB   828          CVTLF   (R5)+,R0                        ; Augment summary buff for this element
                83   50   40  03BE   829          ADDF2   R0,(R3)+                        ; ... with this file's contribution from SYS
                  F7 54   F5  03C1   830          SOBGTR  R4,57$                          ; Loop to pick up all SYSTEM items
                                03C4   831
                   0028   31  03C4   832          BRW     95$                             ; skip to end
                                03C7   833 ;
                                03C7   834 ; Fill summary buff for Heterogeneous classes other than SYSTEM
                                03C7   835 ;
                                03C7   836 70$:
            55   2E A6   D0  03C7   837          MOVL    CDB$A_BUFFERS(R6),R5    ; Get addr of SUM buffer
            55   14 A5   D0  03CB   838          MOVL    MBP$A_SUM(R5),R5        ; ... for this input file
                                03CF   839
                                03CF   840 ;
                                03CF   841 ; Now loop once for each element, moving or adding data from
                                03CF   842 ; the SUM buffer to the m.f. summary buffer.
                                03CF   843 ;
                                03CF   844
                                03CF   845
            54   18 A6   D0  03CF   846          MOVL    CDB$L_ECOUNT(R6),R4     ; Get no. of elements for this class
        0E 43 A7    0C   E0  03D3   847          BBS     #MRB$V_BY_NODE,MRB$W_FLAGS(R7),90$ ; Go do a sum if by node
                                03D8   848 80$:
                83   85   4E  03D8   849          CVTLF   (R5)+,(R3)+                     ; Fill summary buffer for this element
                  FA 54   F5  03DB   850          SOBGTR  R4,80$                          ; Loop to do entire column
                                03DE   851
                83   51   D0  03DE   852          MOVL    R1,(R3)+                        ; Also move in floating seconds
                63   52   D0  03E1   853          MOVL    R2,(R3)                         ; ... and collection count
                     0F   11  03E4   854          BRB     CS_NORMAL                       ; ... and go return
                                03E6   855 90$:
                50   85   4E  03E6   856          CVTLF   (R5)+,R0                        ; Augment summary buff for this element
                83   50   40  03E9   857          ADDF2   R0,(R3)+                        ; ... with this file's contribution
                  F7 54   F5  03EC   858          SOBGTR  R4,90$                          ; Loop to do entire column
                                03EF   859
                                03EF   860 95$:
                83   51   40  03EF   861          ADDF2   R1,(R3)+                        ; Also add in floating seconds
```

```
                63   52   40  03F2   862          ADDF2   R2,(R3)                ; ... and collection count
                          03F5   863
                          03F5   864 CS_NORMAL:
          50  00000000'8F   D0  03F5   865          MOVL    #SS$_NORMAL,R0       ; Indicate success
                          03FC   866 CS_RET:
                          04  03FC   867          RET                          ; ... and return to caller
                              03FD   868
```

```
                                                N 1
03FD    870          .SBTTL  FILL_HOM_SUMMBUFF - Move SUM buffers to Summary Buffer (Homog)
03FD    871
03FD    872 ;++
03FD    873 ;
03FD    874 ; FUNCTIONAL DESCRIPTION:
03FD    875 ;
03FD    876 ;       Augment multi-file summary buffer for the current (homogeneous)
03FD    877 ;       class from the current SUM buffer. Do it for all elements and items.
03FD    878 ;
03FD    879 ; CALLING SEQUENCE:
03FD    880 ;
03FD    881 ;       CALLS   #4,FILL_HOM_SUMMBUFF
03FD    882 ;
03FD    883 ; INPUTS:
03FD    884 ;
03FD    885 ;        4(AP) - pointer to the CDB (Class Descriptor Block)
03FD    886 ;                for the class to process.
03FD    887 ;
03FD    888 ;        8(AP) - address of the first byte of the portion of
03FD    889 ;                the summary buffer for the requested column
03FD    890 ;                for the first item.
03FD    891 ;
03FD    892 ;        12(AP) - floating number of seconds for this input file.
03FD    893 ;
03FD    894 ;        16(AP) - floating number of collections for this input file.
03FD    895 ;
03FD    896 ; IMPLICIT INPUTS:
03FD    897 ;
03FD    898 ;
03FD    899 ; OUTPUTS:
03FD    900 ;
03FD    901 ;       None
03FD    902 ;
03FD    903 ; IMPLICIT OUTPUTS:
03FD    904 ;
03FD    905 ;       CDX$L_DCOUNT -- Count of elements in Super Element ID Table.
03FD    906 ;                       Updated to include any new elements from
03FD    907 ;                       current input file.
03FD    908 ;
03FD    909 ; ROUTINE VALUE:
03FD    910 ;
03FD    911 ;       R0 = SS$_NORMAL, or failing MONITOR status
03FD    912 ;
03FD    913 ; SIDE EFFECTS:
03FD    914 ;
03FD    915 ;       None
03FD    916 ;
03FD    917 ; REGISTER USAGE:
03FD    918 ;
03FD    919 ;       R0-R3 = scratch
03FD    920 ;       R4 = super element ID table index
03FD    921 ;       R5 = super element ID table size (in longwords)
03FD    922 ;       R6 = address of CDB for this class
03FD    923 ;       R7 = address of CDX for this class
03FD    924 ;       R8 = address of current element in element ID table
03FD    925 ;       R9 = address of TM3 temp area
03FD    926 ;       R10 = length of super element ID
```

```
                                                     B 2
                     03FD   927 ;       R11 = address of current element in super element ID table
                     03FD   928 ;
                     03FD   929 ;--
                     03FD   930
             0FFC    03FD   931 .ENTRY  FILL_HOM_SUMMBUFF, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     03FF   932
 56   04 AC    D0    03FF   933         MOVL    4(AP),R6                    ; Get ptr to CDB
 57   32 A6    D0    0403   934         MOVL    CDB$A_CDX(R6),R7            ; Load CDX addr
                     0407   935         ALLOC   TM3$K_SIZE,R0,R9            ; Allocate local temp storage
 5A   09 A7    9A    0414   936         MOVZBL  CDX$B_ELIDLEN(R7),R10       ; Get length of an element ID
                     0418   937
```

```
                     0418   939 ;
                     0418   940 ; Loop through all elements in the element ID table. There is one element
                     0418   941 ; in the table for each element in the SUM buffer (there are CDB$L_ECOUNT
                     0418   942 ; elements in the table). For each element, try to find a match in the
                     0418   943 ; super element ID table. There is one element in this table for each
                     0418   944 ; element in the m.f. summary buffer. The super ID table contains a
                     0418   945 ; number of elements (CDB$L_DCOUNT) which is the accumulation of
                     0418   946 ; distinct elements from all input files for this multi-file
                     0418   947 ; summary request. On the first time through this routine, the
                     0418   948 ; table will be empty.
                     0418   949 ;
                     0418   950
     58    0C A7   D0 0418   951          MOVL    CDX$A_ELIDTABLE(R7),R8  ; Load input Element ID Table addr
           69   D4   041C   952          CLRL    TM3$L_INPIDX(R9)        ; Clear input elidtable index
                     041E   953 10$:
     5B    18 A7   D0 041E   954          MOVL    CDX$A_SELIDTABLE(R7),R11 ; Load Super Element ID Table addr
           14 A9   94 0422   955          CLRB    TM3$B_FOUND(R9)         ; Clear "element found" indicator
              54   D4 0425   956          CLRL    R4                      ; Clear Super Element ID Table index
     55    1C A7   D0 0427   957          MOVL    CDX$L_DCOUNT(R7),R5     ; Load no. of elements in Super ID table
              2A   13 042B   958          BEQL    40$                     ; Br if table is empty
        57    5A   D0 042D   959          MOVL    R10,R7                  ; Borrow R7 to hold elt id length
  0F 4B A6    06   E1 0430   960          BBC     #CDB$V_DISKAC, -        ; Branch if no allocation class in name
                     0435   961                  CDB$L_FLAGS(R6),20$     ; ...
  03 4B A6    07   E1 0435   962          BBC     #CDB$V_DISKVN, -        ; Branch if no volume name in disk name
                     043A   963                  CDB$L_FLAGS(R6),15$     ; ...
        57    0C   C2 043A   964          SUBL2   #12,R7                  ; Shorten length for compare so
                     043D   965                                          ; ... volume name will not be compared
                     043D   966 15$:
              68   95 043D   967          TSTB    (R8)                    ; Allocation class zero ?
              03   13 043F   968          BEQL    20$                     ; Br if so
        57    08   C2 0441   969          SUBL2   #8,R7                   ; Shorten length for compare so
                     0444   970                                          ; ... node name will not be compared
                     0444   971 20$:
  6B  68    57   29 0444   972          CMPC3   R7,(R8),(R11)           ; Match current element in table ?
              06   12 0448   973          BNEQU   30$                     ; Br if not
     14 A9    01   90 044A   974          MOVB    #1,TM3$B_FOUND(R9)      ; Yes -- indicate so
              07   11 044E   975          BRB     40$                     ; ... and terminate loop
                     0450   976 30$:
        5B    5A   C0 0450   977          ADDL2   R10,R11                 ; Point to next Super Element ID
     ED 54    55   F2 0453   978          AOBLSS  R5,R4,20$               ; Loop through Super Element ID Table
                     0457   979
                     0457   980 ;
                     0457   981 ; At this point the entire Super Element ID Table has been scanned for a
                     0457   982 ; match to the current element in the Element ID Table.
                     0457   983 ;
                     0457   984
                     0457   985 40$:
     57    32 A6   D0 0457   986          MOVL    CDB$A_CDX(R6),R7        ; Re-load CDX addr
     04 A9    54   D0 045B   987          MOVL    R4,TM3$L_SUPIDX(R9)     ; Save index for Super Table element
        11 14 A9   E8 045F   988          BLBS    TM3$B_FOUND(R9),70$     ; Branch if element found in table
                     0463   989 ;
                     0463   990 ; Element in input Element ID Table was NOT found in Super Element ID Table.
                     0463   991 ;
                     0463   992
                     0463   993
              21   10 0463   994          BSBB    CHECK_TAB_SPACE         ; Check if table space exhausted
           1D 50   E9 0465   995          BLBC    R0,FHSB_RET             ; If so, go exit with error
```

0468 996

```
                                0468   998 ;
                                0468   999 ; Add a new element to the Super Element ID Table.
                                0468  1000 ;
                                0468  1001
              1C A7   D6        0468  1002          INCL    CDX$L_DCOUNT(R7)            ; Count the new element
          54   5A   C4        046B  1003          MULL2   R10,R4                      ; Compute offset to new table entry
18 B744   68   5A   28        046E  1004          MOVC3   R10,(R8),@CDX$A_SELIDTABLE(R7)[R4]
                                0474  1005                                              ; Move new element ID into the table
                                0474  1006                                              ; NOTE -- R0-R5 destroyed
                                0474  1007
                                0474  1008 ;
                                0474  1009 ; Move entry for each item (for this element)
                                0474  1010 ; from SUM buffer to M.F. Summary Buffer.
                                0474  1011 ;
                                0474  1012
                                0474  1013 70$:
                                0474  1014
                     35   10    0474  1015          BSBB    SUM_TO_SUMMBUFF             ; Augment summary buff from SUM buff
                                0476  1016                                              ; NOTE -- destroys regs R0-R5, R11
                                0476  1017
                58   5A   C0    0476  1018          ADDL2   R10,R8                      ; Point to next input element
                                0479  1019
     A0 69    18 A6   F2        0479  1020          AOBLSS  CDB$L_ECOUNT(R6), -         ; Loop once for each input element
                                047E  1021                  TM3$L_INPIDX(R9),10$
                                047E  1022
 50   00000000'8F   D0        047E  1023          MOVL    #SS$_NORMAL,R0              ; Indicate success
                                0485  1024
                                0485  1025 FHSB_RET:
                     04         0485  1026          RET                                 ; ... and return to caller
                                0486  1027
                                0486  1028
                                0486  1029 CHECK_TAB_SPACE:
                                0486  1030
 50   00000000'8F   D0        0486  1031          MOVL    #SS$_NORMAL,R0              ; Assume normal status
 54   00000000'8F   D1        048D  1032          CMPL    #MAXELTS_MFS,R4             ; Have we run out of table space?
                14   14        0494  1033          BGTR    10$                         ; Br if not
      00000000'8F   DD        0496  1034          PUSHL   #MNR$_TABLEFULL             ; Stack MONITOR failing status code
00000000'EF   01   FB        049C  1035          CALLS   #1,MON_ERR                  ; Log the error
 50   00000000'8F   D0        04A3  1036          MOVL    #MNR$_TABLEFULL,R0          ; Get status to caller
                                04AA  1037 10$:
                     05         04AA  1038          RSB                                 ; Return to caller
```

```
04AB  1040          .SBTTL  SUM_TO_SUMMBUFF - Move SUM buffer to Summary Buffer
04AB  1041
04AB  1042 ;++
04AB  1043 ;
04AB  1044 ; FUNCTIONAL DESCRIPTION:
04AB  1045 ;
04AB  1046 ;       This subroutine augments the multi-file summary buffer for
04AB  1047 ;       the current (homogeneous) class with information from the
04AB  1048 ;       SUM buffer for a single element.
04AB  1049 ;
04AB  1050 ; CALLING SEQUENCE:
04AB  1051 ;
04AB  1052 ;       BSBW    SUM_TO_SUMMBUFF
04AB  1053 ;
04AB  1054 ; INPUTS:
04AB  1055 ;
04AB  1056 ;       R0-R5 and R11 = scratch
04AB  1057 ;       R6 = address of CDB for current class
04AB  1058 ;       R7 = address of CDX for current class
04AB  1059 ;       R9 = address of TM3 (local temporary storage block)
04AB  1060 ;
04AB  1061 ; IMPLICIT INPUTS:
04AB  1062 ;
04AB  1063 ;       TM3$L_INPIDX = index of desired element in SUM buffer
04AB  1064 ;       TM3$L_SUPIDX = index of desired element in Summary Buffer
04AB  1065 ;       See INPUTs for routine which calls this one (FILL_HOM_SUMMBUFF).
04AB  1066 ;
04AB  1067 ; OUTPUTS:
04AB  1068 ;
04AB  1069 ;       None
04AB  1070 ;
04AB  1071 ; IMPLICIT OUTPUTS:
04AB  1072 ;
04AB  1073 ;       The M.F. Summary Buffer is augmented for the current element of
04AB  1074 ;       the current (homogeneous class).
04AB  1075 ;
04AB  1076 ; ROUTINE VALUE:
04AB  1077 ;
04AB  1078 ;       None
04AB  1079 ;
04AB  1080 ; SIDE EFFECTS:
04AB  1081 ;
04AB  1082 ;       None
04AB  1083 ;
04AB  1084 ; REGISTER USAGE:
04AB  1085 ;
04AB  1086 ;       R0 = scratch
04AB  1087 ;       R1 = address of destination longword for seconds
04AB  1088 ;       R2 = index into input SUM buffer for this element
04AB  1089 ;       R3 = index into output summary buffer for this element
04AB  1090 ;       R4 = address of destination longword
04AB  1091 ;       R5 = address of source longword
04AB  1092 ;       R6 = address of CDB for this class
04AB  1093 ;       R7 = address of CDX for this class
04AB  1094 ;       R8 = unused
04AB  1095 ;       R9 = address of TM3 temp area
04AB  1096 ;       R10 = unused
```

```
                                04AB  1097 ;       R11 = item index
                                04AB  1098 ;
                                04AB  1099 ;--
                                04AB  1100
                                04AB  1101 SUM_TO_SUMMBUFF:
                                04AB  1102
          0C A9   08 AC   D0    04AB  1103         MOVL    8(AP),TM3$A_SBCOL(R9)   ; Get addr of portion of summary
                                04B0  1104                                         ; ... buffer for this column
              52   69   D0    04B0  1105         MOVL    TM3$L_INPIDX(R9),R2     ; Get SUM buffer index
           53   04 A9   D0    04B3  1106         MOVL    TM3$L_SUPIDX(R9),R3     ; Get summary buffer index
          08 A9   06 A7   9A    04B7  1107         MOVZBL  CDX$B_IDISCT(R7), -     ; Get # of items to do
                                04BC  1108                 TM3$L_ITEMS(R9)         ;
      10 A9   08 A6   08 A9   C7    04BC  1109         DIVL3   TM3$L_ITEMS(R9), -      ; Compute len of summary buff for 1 item
                                04C3  1110                 CDB$L_SUMBUF(R6),TM3$L_SBLEN(R9)
                 5B   D4    04C3  1111         CLRL    R11                     ; Clear item index
                                04C5  1112 10$:
            54   0C B943   DE    04C5  1113         MOVAL   @TM3$A_SBCOL(R9)[R3],R4 ; Compute addr of destination longword
            50   2E B64B   D0    04CA  1114         MOVL    @CDB$A_BUFFERS(R6)[R11],R0 ; Get addr of MBP for this item
            55   14 B042   DE    04CF  1115         MOVAL   @MBP$A_SUM(R0)[R2],R5   ; Compute addr of source longword
        50    00000000'EF   D0    04D4  1116         MOVL    MRBPTR,R0               ; Get MRB ptr
          05 43 A0   0C     E0    04DB  1117         BBS     #MRB$V_BY_NODE,MRB$W_FLAGS(R0),20$ ; Go do a sum if by node
                 64   65   4E    04E0  1118         CVTLF   (R5),(R4)               ; Move SUM element to summary buffer
                 06   11   04E3  1119         BRB     30$                     ; Go check seconds and colls
                                04E5  1120 20$:
              50   65   4E    04E5  1121         CVTLF   (R5),R0                 ; Convert SUM element to floating
              64   50   40    04E8  1122         ADDF2   R0,(R4)                 ; ... and add into summary buffer
                                04EB  1123 30$:
                 52   D5    04EB  1124         TSTL    R2                      ; Is this the first SUM buffer element?
                 29   12    04ED  1125         BNEQ    50$                     ; No -- skip secs and colls processing
51    0C A9   00000000'8F   C1    04EF  1126         ADDL3   #<4*MAXELTS_MFS>, -     ; Yes -- Get ptr to destination for secs
                                04F8  1127                 TM3$A_SBCOL(R9),R1
          50    00000000'EF   D0    04F8  1128         MOVL    MRBPTR,R0               ; Re-load MRB ptr
             0B 43 A0   0C     E0    04FF  1129         BBS     #MRB$V_BY_NODE,MRB$W_FLAGS(R0),40$ ; Go do a sum if by node
                61   0C AC   D0    0504  1130         MOVL    12(AP),(R1)             ; Get seconds into summary buffer
             04 A1   10 AC   D0    0508  1131         MOVL    16(AP),4(R1)            ; Get collections into summary buffer
                 09   11   050D  1132         BRB     50$                     ; Go process next item
                                050F  1133 40$:
                61   0C AC   40    050F  1134         ADDF2   12(AP),(R1)             ; Add in seconds
             04 A1   10 AC   40    0513  1135         ADDF2   16(AP),4(R1)            ; ... and collections
                                0518  1136 50$:
             0C A9   10 A9   C0    0518  1137         ADDL2   TM3$L_SBLEN(R9), -      ; Point to this column's summary
                                051D  1138                 TM3$A_SBCOL(R9)         ; ... buffer for next item
                                051D  1139
             A3 5B   08 A9   F2    051D  1140         AOBLSS  TM3$L_ITEMS(R9),R11,10$ ; Loop to process next requested item
                                0522  1141
                                0522  1142         RSB                             ; Return to caller
```

```
0523  1144          .SBTTL  ALLOC_SUMBUFS - Allocate Summary Buffers
0523  1145 ;++
0523  1146 ;
0523  1147 ; FUNCTIONAL DESCRIPTION:
0523  1148 ;
0523  1149 ;       This routine is called to allocate and clear to zeroes
0523  1150 ;       the entire Multi-file Summary Buffer for each requested class.
0523  1151 ;       In addition, if the class is homogeneous, allocate the
0523  1152 ;       Super Element ID Table.
0523  1153 ;
0523  1154 ; CALLING SEQUENCE:
0523  1155 ;
0523  1156 ;       CALLS #1,ALLOC_SUMBUFS
0523  1157 ;
0523  1158 ; INPUTS:
0523  1159 ;
0523  1160 ;       4(AP)   - address of MRB$O_CLASSBITS, the octaword
0523  1161 ;                 bit string representing classes needing service.
0523  1162 ;
0523  1163 ; IMPLICIT INPUTS:
0523  1164 ;
0523  1165 ;       None
0523  1166 ;
0523  1167 ; OUTPUTS:
0523  1168 ;
0523  1169 ;       None
0523  1170 ;
0523  1171 ; IMPLICIT OUTPUTS:
0523  1172 ;
0523  1173 ;       Multi-file Summary Buffer for each requested class
0523  1174 ;       (represented by CDB$L_SUMBUF and CDB$A_SUMBUF) allocated
0523  1175 ;       and cleared to zeroes. If homogeneous, super element ID
0523  1176 ;       table (represented by CDX$L_SELIDTABLE and CDX$A_SELIDTABLE)
0523  1177 ;       allocated.
0523  1178 ;
0523  1179 ;
0523  1180 ; ROUTINE VALUE:
0523  1181 ;
0523  1182 ;       R0 = SS$_NORMAL, or failing status code from LIB$GET_VM.
0523  1183 ;
0523  1184 ; SIDE EFFECTS:
0523  1185 ;
0523  1186 ;       None
0523  1187 ;
0523  1188 ;--
0523  1189
```

```
                                0FFC  0523  1191 .ENTRY  ALLOC_SUMBUFS,  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                      0525  1192
                                      0525  1193
                          57    D4  0525  1194          CLRL    R7                              ; Init starting bit position
                                0527  1195 20$:
                       59 20    D0  0527  1196          MOVL    #32,R9                          ; Init bit field size
                                052A  1197                                                      ; NOTE -- must handle in 32-bit chunks
                       58 57    D0  052A  1198          MOVL    R7,R8                           ; Init start position of next chunk
                                052D  1199 30$:
     5A  04 BC         59 58    EA  052D  1200          FFS     R8,R9,@4(AP),R10                ; Search for next class number
                                0533  1201                                                      ; R10 contains class no. if found
                          11    13  0533  1202          BEQL    40$                             ; Branch if none found this chunk
                                0535  1203
                          1F    10  0535  1204          BSBB    AS_BUFF                         ; Alloc buffers for this class
                       1B 50    E9  0537  1205          BLBC    R0,AS_ERR                       ; Go return if error
                                053A  1206
                       59 58    C0  053A  1207          ADDL2   R8,R9                           ; Compute next starting
                    58 5A 01    C1  053D  1208          ADDL3   #1,R10,R8                       ; ... position and field size
                    59 58       C2  0541  1209          SUBL2   R8,R9                           ; ... for this chunk
                          E7    11  0544  1210          BRB     30$                             ; Go search rest of chunk
                                0546  1211 40$:
 FFD9 57 20 0000'8F             3D  0546  1212          ACBW    #MAX_CLASS_NO,#32,R7,20$ ; Loop to process next chunk
                                054E  1213
      50 00000000'8F            D0  054E  1214          MOVL    #SS$_NORMAL,R0                  ; Set normal status
                                0555  1215 AS_ERR:
                          04    05  0555  1216          RET                                     ; Return with status in R0
                                0556  1217
                                0556  1218
                                0556  1219 AS_BUFF:                                           ; Allocate buffers for this class
                                0556  1220                                                      ; NOTE -- R10 contains class number
                                0556  1221                                                      ; Regs R0 thru R6 are scratch
                                0556  1222                                                      ; Regs R7 thru R10 must not be changed
                                0556  1223
56 5A 00000053 8F               C5  0556  1224          MULL3   #CDB$K_SIZE,R10,R6              ; Compute offset to desired CDB
   56 00000000'EF46             9E  055E  1225          MOVAB   CDBHEAD[R6],R6                  ; Index to CDB address
                                0566  1226
                    0C A6       D5  0566  1227          TSTL    CDB$A_SUMBUF(R6)                ; Do we have summary buffer yet?
                       03       13  0569  1228          BEQLU   10$                             ; If not, go get it
                     00A5       31  056B  1229          BRW     60$                             ; Otherwise, exit
                                056E  1230
                                056E  1231 ;
                                056E  1232 ; Compute size of summary buffer and issue call to LIB$GET_VM
                                056E  1233 ; to get the space.
                                056E  1234 ;
                                056E  1235
                                056E  1236 10$:
     09 4B A6     08            E1  056E  1237          BBC     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),15$ ; Br if not SYSTEM class
   50 00000002'8F                D0  0573  1238          MOVL    #<ECOUNT_SYS_ALL+2>,R0          ; Get special ECOUNT
                          1A    11  057A  1239          BRB     20$                             ; ... and continue
                                057C  1240 15$:
    50 18 A6     02               C1  057C  1241          ADDL3   #2,CDB$L_ECOUNT(R6),R0          ; Compute size of summ buff for 1 col.
  10 4B A6       05               E1  0581  1242          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),20$ ; Br if a hetero class
       51   32 A6                 D0  0586  1243          MOVL    CDB$A_CDX(R6),R1                ; Get CDX address
       51   06 A1                 9A  058A  1244          MOVZBL  CDX$B_IDISCT(R1),R1             ; Get number of items
50 51 00000002'8F                  C5  058E  1245          MULL3   #<MAXELTS_MFS+2>,R1,R0          ; Compute size for homog class
                                0596  1246 20$:
                       50 04    C4  0596  1247          MULL2   #4,R0                           ; Translate lwords to bytes
```

```
                51   00000000'EF   D0  0599  1248          MOVL    MFSPTR,R1                       ; Get MFS ptr
                  51     3A A1     9A  05A0  1249          MOVZBL  MFS$B_COLUMNS(R1),R1            ; Get number of columns
                08 A6    51   50   C5  05A4  1250          MULL3   R0,R1,CDB$L_SUMBUF(R6)          ; Compute size of entire summary buffer
                                     05A9  1251
                         0C A6   DF  05A9  1252          PUSHAL  CDB$A_SUMBUF(R6)                ; Push addr of longword to hold
                                     05AC  1253                                                  ; ... summary buffer pointer
                         08 A6   DF  05AC  1254          PUSHAL  CDB$L_SUMBUF(R6)                ; Push addr of # of bytes needed
              00000000'GF    02   FB  05AF  1255          CALLS   #2,G^LIB$GET_VM                 ; Allocate buffers
                         61 50   E9  05B6  1256          BLBC    R0,70$                          ; Leave if error
                                     05B9  1257
                                     05B9  1258 ;
                                     05B9  1259 ; Now get space for the Super Element ID Table for homog class
                                     05B9  1260 ; and initialize CDX$L_DCOUNT, which is used to hold the size of the table.
                                     05B9  1261 ;
                                     05B9  1262
                   24 4B A6    05   E1  05B9  1263          BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),30$ ; Br if a hetero class
                                     05BE  1264
                      50    32 A6   D0  05BE  1265          MOVL    CDB$A_CDX(R6),R0                ; Get CDX address
                            1C A0   D4  05C2  1266          CLRL    CDX$L_DCOUNT(R0)                ; Clear no. of elts in the S ID Tab
                      51    09 A0   9A  05C5  1267          MOVZBL  CDX$B_ELIDLEN(R0),R1            ; Get length of an element ID
          14 A0       51   00000000'8F   C5  05C9  1268          MULL3   #MAXELTS_MFS,R1, -
                                     05D2  1269                  CDX$L_SELIDTABLE(R0)            ; Compute length of Sup Elt ID Table
                                     05D2  1270
                         18 A0   DF  05D2  1271          PUSHAL  CDX$A_SELIDTABLE(R0)            ; Push addr of lword to hold table ptr
                         14 A0   DF  05D5  1272          PUSHAL  CDX$L_SELIDTABLE(R0)            ; Push addr of # of bytes needed
              00000000'GF    02   FB  05D8  1273          CALLS   #2,G^LIB$GET_VM                 ; Allocate table
                         38 50   E9  05DF  1274          BLBC    R0,70$                          ; Leave if error
                                     05E2  1275
                                     05E2  1276 ;
                                     05E2  1277 ; Clear the entire summary buffer for this class to zeroes.
                                     05E2  1278 ;
                                     05E2  1279
                                     05E2  1280 30$:
                      5B    0C A6   D0  05E2  1281          MOVL    CDB$A_SUMBUF(R6),R11            ; Load address of summary buffer
                      56    08 A6   D0  05E6  1282          MOVL    CDB$L_SUMBUF(R6),R6             ; Load size of buffer
                                     05EA  1283 40$:
                56   00007D00   8F   D1  05EA  1284          CMPL    #32000,R6                       ; Is a large MOVC5 required?
                                19   18  05F1  1285          BGEQ    50$                             ; No -- go do a smaller one
6B    7D00 8F     00    FE AF     00   2C  05F3  1286          MOVC5   #0,,#0,#32000,(R11)             ; Yes -- clear 32000 bytes
                56   00007D00   8F   C2  05FC  1287          SUBL2   #32000,R6                       ; Calc bytes left to clear
                5B   00007D00   8F   C0  0603  1288          ADDL2   #32000,R11                      ; ... and starting byte addr
                                DE   11  060A  1289          BRB     40$                             ; Go check size of next move
                                     060C  1290 50$:
     6B    56     00    FE AF     00   2C  060C  1291          MOVC5   #0,,#0,R6,(R11)                 ; Zero out SUMBUF for this class
                                     0613  1292 60$:
                50   00000000'8F   D0  0613  1293          MOVL    #SS$_NORMAL,R0                  ; Success status
                                     061A  1294 70$:
                                05   05  061A  1295          RSB                                     ; Return with status in R0
```

```
                        061B  1297          .SBTTL  MFS_FREE_MEM - Free Virtual Memory for m.f.s.
                        061B  1298
                        061B  1299 ;++
                        061B  1300 ;
                        061B  1301 ; FUNCTIONAL DESCRIPTION:
                        061B  1302 ;
                        061B  1303 ;       This routine issues calls to LIB$FREE_VM to free up virtual
                        061B  1304 ;       memory acquired by classes for multi-file summary buffers and
                        061B  1305 ;       (for homogeneous classes) super element ID tables. No status
                        061B  1306 ;       code checking is done, since this routine is in a cleanup path.
                        061B  1307 ;
                        061B  1308 ; INPUTS:
                        061B  1309 ;
                        061B  1310 ;       None
                        061B  1311 ;
                        061B  1312 ; IMPLICIT INPUTS:
                        061B  1313 ;
                        061B  1314 ;       The CDB$L_SUMBUF, CDB$A_SUMBUF, CDX$L_SELIDTABLE and CDX$A_SELIDTABLE
                        061B  1315 ;       fields contain the length and address, respectively, of memory
                        061B  1316 ;       blocks to be freed (for each class in this MONITOR request).
                        061B  1317 ;
                        061B  1318 ;       MFSPTR  -  pointer to Multi-File Summary Block
                        061B  1319 ;
                        061B  1320 ; OUTPUTS:
                        061B  1321 ;
                        061B  1322 ;       None
                        061B  1323 ;
                        061B  1324 ; IMPLICIT OUTPUTS:
                        061B  1325 ;
                        061B  1326 ;       Memory is freed. Pointers to freed memory are cleared to 0.
                        061B  1327 ;
                        061B  1328 ; ROUTINE VALUE:
                        061B  1329 ;
                        061B  1330 ;       NORMAL
                        061B  1331 ;
                        061B  1332 ; SIDE EFFECTS:
                        061B  1333 ;
                        061B  1334 ;       None
                        061B  1335 ;
                        061B  1336 ;--
                        061B  1337
                  01FC  061B  1338 .ENTRY  MFS_FREE_MEM, ^M<R2,R3,R4,R5,R6,R7,R8>
                        061D  1339
57   00000000'EF    D0  061D  1340          MOVL    MFSPTR,R7                       ; Load MFS pointer
                55    D4  0624  1341          CLRL    R5                              ; Init starting bit position
                        0626  1342 20$:
            53  20    D0  0626  1343          MOVL    #32,R3                          ; Init bit field size
                        0629  1344                                                  ; NOTE -- must handle in 32-bit chunks
            52  55    D0  0629  1345          MOVL    R5,R2                           ; Init start position of next chunk
                        062C  1346 30$:
54  67  53  52    EA  062C  1347          FFS     R2,R3,MFS$O_CLASSBITS(R7),R4 ; Search for next class number
                        0631  1348                                                  ; R4 contains class no. if found
                0E  13  0631  1349          BEQL    40$                             ; Branch if none found this chunk
                1C  10  0633  1350          BSBB    FREE_CLASS                      ; Free memory for this class
            53  52    C0  0635  1351          ADDL2   R2,R3                           ; Compute next starting
        52  54  01    C1  0638  1352          ADDL3   #1,R4,R2                        ; ... position and field size
            53  52    C2  063C  1353          SUBL2   R2,R3                           ; ... for this chunk
```

```
                     EB   11  063F  1354                 BRB     30$                             ; Go search rest of chunk
                                0641  1355 40$:
         FFDD 55   20   0000'8F  3D  0641  1356                 ACBW    #MAX_CLASS_NO,#32,R5,20$ ; Loop to process next chunk
                                0649  1357
              50   00000000'EF  D0  0649  1358                 MOVL    NORMAL,R0                ; Set normal status
                                04  0650  1359                 RET                              ; Return
                                    0651  1360
                                    0651  1361 FREE_CLASS:                                     ; Free class memory
                                    0651  1362                                                 ; NOTE -- R4 contains class number
                                    0651  1363
           56   54   00000053 8F  C5  0651  1364                 MULL3   #CDB$K_SIZE,R4,R6        ; Compute offset to desired CDB
                56   00000000'EF46  9E  0659  1365                 MOVAB   CDBHEAD[R6],R6           ; Index to CDB address
                                    0661  1366
                          0C A6  D5  0661  1367                 TSTL    CDB$A_SUMBUF(R6)         ; Is there a m.f. summary buffer?
                             10  13  0664  1368                 BEQL    20$                      ; Branch if not
                          0C A6  DF  0666  1369                 PUSHAL  CDB$A_SUMBUF(R6)         ; Stack addr of buffer ptr
                          08 A6  DF  0669  1370                 PUSHAL  CDB$L_SUMBUF(R6)         ; Stack addr of buffer length
                00000000'GF   02  FB  066C  1371                 CALLS   #2,G^LIB$FREE_VM         ; Free it
                          0C A6  D4  0673  1372                 CLRL    CDB$A_SUMBUF(R6)         ; Clear address
                                0676  1373 20$:
                    19 4B A6   05  E1  0676  1374                 BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),30$ ; Br if a hetero class
                       58   32 A6  D0  067B  1375                 MOVL    CDB$A_CDX(R6),R8         ; Get CDX address
                          18 A8  D5  067F  1376                 TSTL    CDX$A_SELIDTABLE(R8)     ; Is there a super element ID tab?
                             10  13  0682  1377                 BEQL    30$                      ; Branch if not
                          18 A8  DF  0684  1378                 PUSHAL  CDX$A_SELIDTABLE(R8)     ; Stack addr of table
                          14 A8  DF  0687  1379                 PUSHAL  CDX$L_SELIDTABLE(R8)     ; Stack addr of table length
                00000000'GF   02  FB  068A  1380                 CALLS   #2,G^LIB$FREE_VM         ; Free it
                          18 A8  D4  0691  1381                 CLRL    CDX$A_SELIDTABLE(R8)     ; Clear address
                                0694  1382 30$:
                             05  0694  1383                 RSB                              ; Return
                                0695  1384
                                0695  1385 .END
```

```
ALLOC_SUMBUFS           00000523 RG    03
ALL_STAT              = 00000000
AS_BUFF                 00000556 R     03
AS_ERR                  00000555 R     03
AVE_STAT              = 00000002
BLANK_STR               0000016F R     01
BLANK_STR1              0000017C R     01
CAPTURE_SUMS            000002D4 RG    03
CDB                   = 00000000
CDB$A_BUFFERS         = 0000002E
CDB$A_CDX             = 00000032
CDB$A_CHDHDR          = 0000004F
CDB$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            = 0000000B
CDB$W_BLKLEN          = 00000020
CDB$W_DISPCTL         = 00000036
CDB$W_QFLAGS          = 00000045
CDB$W_QFLAGS_CUR      = 00000049
CDB$W_QFLAGS_DEF      = 00000047
CDBHEAD                 ******** X   03
CDB_EXT               = 00000000
CDX$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$W_IBITS_DEF       = 00000002
CHECK_TAB_SPACE         00000486 R     03
COMPUTED_ITEM_AVE       00000158 R     03
COUNT_TYPE              ******** X   03
CSB                   = 00000000
CSB$B_FILES           = 00000021
CSB$B_FLAGS           = 00000020
CSB$K_SIZE            = 00000022
CSB$Q_BEGINNING       = 00000010
CSB$Q_ENDING          = 00000018
CSB$S_BEGINNING       = 00000008
CSB$S_CSB             = 00000022
CSB$S_ENDING          = 00000008
CSB$S_FILLER          = 00000007
CSB$S_FLAGS           = 00000001
CSB$S_NODENAME        = 00000010
CSB$T_NODENAME        = 00000000
CSB$V_FILLER          = 00000001
CSB$V_IGNORE          = 00000000
```

```
CSBVEC_PTR                      ********    X   03
CS_NORMAL                       000003F5 R      03
CS_RET                          000003FC R      03
CUR_STAT                    =   00000001
CVTSTK                          0000017C R      03
CVTSTK10                        00000197 R      03
DATA_STR                        00000147 R      01
DATA_STR1                       0000015D R      01
ECOUNT_SYS_ALL                  ********    X   03
ESC                         =   0000001B
ESTAB_FAOSTR                    0000021C R      03
FAOSTK                          ********    X   03
FHSB_RET                        00000485 R      03
FILL_HOM_SUMMBUFF               000003FD RG     03
FILL_MFSOM_FAOSTK               00000000 RG     03
FMF_ELEM                        000000AF R      03
FMF_RET                         00000150 R      03
FREE_CLASS                      00000651 R      03
IDB                         =   00000000
IDB$A_ADDR                  =   0000000C
IDB$A_LNAME                 =   00000004
IDB$A_SNAME                 =   00000000
IDB$B_FLAGS                 =   00000010
IDB$K_ILENGTH               =   00000011
IDB$M_PCNT                  =   00000001
IDB$S_FILLER                =   00000007
IDB$S_FLAGS                 =   00000001
IDB$S_IDB                   =   00000011
IDB$V_FILLER                =   00000001
IDB$V_PCNT                  =   00000000
IDB$W_ISIZE                 =   00000008
IDB$W_TYPE                  =   0000000A
ITMSTR_SYS_ALL                  ********    X   03
LIB$FREE_VM                     ********    X   03
LIB$GET_VM                      ********    X   03
MAXELTS_MFS                     ********    X   03
MAX_CLASS_NO                    ********    X   03
MAX_STAT                    =   00000004
MBP                         =   00000000
MBP$A_ADDR                  =   00000018
MBP$A_B1ST                  =   00000004
MBP$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_STATS                 =   00000008
MBP$A_SUM                   =   00000014
MBP$K_SIZE                  =   00000028
MBP$S_MBP                   =   00000028
MBP$S_MBP2                  =   0000001C
MBP$S_MBP3                  =   0000000C
MBP2                        =   00000000
MBP3                        =   00000000
MCA                         =   00000000
MCA$A_INPUT_PTR             =   00000004
MCA$A_MPADDR                =   0000001C
MCA$B_FIRSTC                =   00000030
MCA$B_LASTC                 =   00000031
MCA$K_SIZE                  =   0000003A
MCA$L_COLLCNT               =   0000000C
MCA$L_CONSEC_REC            =   00000034
MCA$L_DISPCNT               =   00000010
MCA$L_INPUT_LEN             =   00000000
MCA$L_INTTICKS              =   00000008
MCA$L_INT_MULT              =   00000014
MCA$L_PROC_DISP             =   00000018
MCA$Q_CURR_TIME             =   00000020
MCA$Q_LASTCOLL              =   00000028
MCA$S_CURR_TIME             =   00000008
MCA$S_FILLER                =   00000006
MCA$S_FLAGS                 =   00000002
MCA$S_LASTCOLL              =   00000008
MCA$S_MCA                   =   0000003A
MCA$V_ENTRY                 =   00000000
MCA$V_EOF                   =   00000003
MCA$V_ERA_SCRL              =   00000006
MCA$V_FILLER                =   0000000A
MCA$V_FUTURE                =   00000001
MCA$V_GRAPHICS              =   00000005
MCA$V_MULTFND               =   00000002
MCA$V_REFRESH               =   00000008
MCA$V_S_TOP_DISP            =   00000009
MCA$V_TOP_DISP              =   00000007
MCA$V_VIDEO                 =   00000004
MCA$W_DCLASSCT              =   00000038
MCA$W_FLAGS                 =   00000032
MCAPTR                          ********    X   03
MFS                         =   00000000
MFS$A_IFB_TAB               =   00000028
MFS$A_STATSBUF              =   00000024
MFS$A_SUMMARY               =   0000002C
MFS$B_COLUMNS               =   0000003A
MFS$B_CUR_COL               =   0000003B
MFS$B_DATA_COLS             =   0000003C
MFS$K_SIZE                  =   0000003D
MFS$L_ELEMS                 =   00000030
MFS$L_LWORDS                =   00000034
MFS$L_STATSBUF              =   00000020
MFS$O_CLASSBITS             =   00000000
MFS$Q_BEGINNING             =   00000010
MFS$Q_ENDING                =   00000018
MFS$S_BEGINNING             =   00000008
```

```
MFS$S_CLASSBITS                 = 00000010
MFS$S_ENDING                    = 00000008
MFS$S_MFS                       = 0000003D
MFS$W_CLASSCT                   = 00000038
MFSPTR                            ********    X   03
MFSUMSTR                          00000000 RG     02
MFS_FREE_MEM                      0000061B RG     03
MFS_NODE_STR                      00000052 RG     01
MFS_NOD_SEGLEN                  = 00000016   G
MFS_STHEAD1_STR                   000000F9 RG     01
MFS_STHEAD2_STR                   0000011E RG     01
MFS_TIME_STR                      000000C1 RG     01
MFS_TIM_SEGLEN                  = 0000000B   G
MF_STATHEAD_STR                   00000030 RG     01
MF_SUMM1_STR                      00000000 RG     01
MIN_STAT                        = 00000003
MNR$_TABLEFULL                    ********    X   03
MODES_CLSNO                       ********    X   03
MON_ERR                           ********    X   03
MRB                             = 00000000
MRB$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
MRBPTR                            ********    X   03
NORMAL                            ********    X   03
PERFTABLE                         ********    X   03
PROCDISPS                       = 00000005
REG_PROC                        = 00000000
SS$_NORMAL                        ********    X   03
STACK_STATS                       000001DE R      03
STATS                           = 00000005
STATS_STR                         00000189 R      01
SUM_TO_SUMMBUFF                   000004AB R      03
TEMP_2_BLOCK                    = 00000000
TEMP_3_BLOCK                    = 00000000
TM2$R_SIZE                      = 0000001E
TM2$L_COLLS                     = 00000014
TM2$L_COLS                      = 00000004
TM2$L_COLS_USED                 = 0000000C
TM2$L_COL_SIZE                  = 00000008
TM2$L_ELEMS                     = 00000018
TM2$L_SECONDS                   = 00000010
TM2$L_START_COL                 = 00000000
TM2$S_TEMP_2_BLOCK              = 0000001E
TM2$W_ITEM_TYPE                 = 0000001C
TM3$A_SBCOL                     = 0000000C
TM3$B_FOUND                     = 00000014
TM3$K_SIZE                      = 00000015
TM3$L_INPIDX                    = 00000000
TM3$L_ITEMS                     = 00000008
TM3$L_SBLEN                     = 00000010
TM3$L_SUPIDX                    = 00000004
TM3$S_TEMP_3_BLOCK              = 00000015
TOPB_PROC                       = 00000003
TOPC_PROC                       = 00000001
TOPD_PROC                       = 00000002
TOPF_PROC                       = 00000004
UPD_STATS                         000001AE 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
MONRODATA	000001B9 (441.)	01 (1.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC QUAD
MONDATA	00000080 (128.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC QUAD
$$MONCODE	00000695 (1685.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.96
Command processing	108	00:00:00.77	00:00:03.49
Pass 1	200	00:00:04.89	00:00:11.82
Symbol table sort	0	00:00:00.52	00:00:00.71
Pass 2	244	00:00:03.01	00:00:06.90
Symbol table output	39	00:00:00.27	00:00:01.90
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	625	00:00:09.56	00:00:25.84

The working set limit was 1500 pages.
32415 bytes (64 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 334 non-local and 85 local symbols.
1385 source lines were read in Pass 1, producing 33 object records in Pass 2.
16 pages of virtual memory were used to define 11 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MONTOR.OBJ]MONLIB.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	10

299 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

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

MONMSG
LIS

REQUEST
LIS

SHODEF
LIS

MONSUB
LIS

PREPOST
LIS

SUMMBUFF
LIS

BINDVL
LIS

MOUNT

SYSFAO
LIS

MOUNTSHR
MAP

TEMPLATE
LIS

VMOUNT
MAP

ASSIST
LIS

ALLOCM
LIS

MOUDEF
B32