```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```

```
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  MM      MM  EEEEEEEEEE  MM      MM    000000    RRRRRRRR    YY      YY
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  MM      MM  EEEEEEEEEE  MM      MM    000000    RRRRRRRR    YY      YY
GG          EE              TT      MMMM  MMMM  EE          MMMM  MMMM  00      00  RR      RR  YY      YY
GG          EE              TT      MMMM  MMMM  EE          MMMM  MMMM  00      00  RR      RR  YY      YY
GG          EE              TT      MM  MM  MM  EE          MM  MM  MM  00      00  RR      RR    YY  YY
GG          EE              TT      MM  MM  MM  EE          MM  MM  MM  00      00  RR      RR    YY  YY
GG          EEEEEEEE        TT      MM      MM  EEEEEEEE    MM      MM  00      00  RRRRRRRR        YY
GG          EEEEEEEE        TT      MM      MM  EEEEEEEE    MM      MM  00      00  RRRRRRRR        YY
GG  GGGGGG  EE              TT      MM      MM  EE          MM      MM  00      00  RR  RR          YY
GG  GGGGGG  EE              TT      MM      MM  EE          MM      MM  00      00  RR  RR          YY
GG      GG  EE              TT      MM      MM  EE          MM      MM  00      00  RR    RR        YY
GG      GG  EE              TT      MM      MM  EE          MM      MM  00      00  RR    RR        YY
  GGGGGG    EEEEEEEEEE      TT      MM      MM  EEEEEEEEEE  MM      MM    000000    RR      RR      YY
  GGGGGG    EEEEEEEEEE      TT      MM      MM  EEEEEEEEEE  MM      MM    000000    RR      RR      YY

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE GETMEMORY (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 ! WRITTEN BY
;    29          0029  1 !       Bert Beander    August, 1980.
;    30          0030  1 !
;    31          0031  1 ! MODULE FUNCTION
;    32          0032  1 !       This module contains the Debugger's Free Memory Manager, i.e. all the
;    33          0033  1 !       routines which initialize the free memory pool and allocate and dealloc-
;    34          0034  1 !       ate memory blocks.  These memory blocks are used for RST entries, Static
;    35          0035  1 !       Address Table entries, and most other descriptors used in the Debugger.
;    36          0036  1 !
;    37          0037  1 ! MODIFIED BY
;    38          0038  1 !       Ping Sager      May, 1982       Add PUSH and POP routines to manage
;    39          0039  1 !                                       releasing the tempory memory blocks
;    40          0040  1 !                                       in levels.
;    41          0041  1 !       Rich Title      May, 1982       Added DBG$NPARSE_ALLOCATE and
;    42          0042  1 !                                       DBG$NEXECUTE_ALLOCATE to support the
;    43          0043  1 !                                       ALLOCATE command.
;    44          0044  1 !
;    45          0045  1
;    46          0046  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    47          0180  1
;    48          0181  1 FORWARD ROUTINE
;    49          0182  1     DBG$CHECK_MEMORY:   NOVALUE,        ! Check the integrity of the memory pool
;    50          0183  1     DBG$COPY_MEMORY,                    ! Make a copy of a memory block in a
;    51          0184  1                                         !       permanent block
;    52          0185  1     DBG$COPY_TEMPMEM,                   ! Make a copy of a memory block in a
;    53          0186  1                                         !       temporary block
;    54          0187  1     DBG$EXPAND_MEMORY,                  ! Expand the free memory pool
;    55          0188  1     DBG$FREE_MEM_LEFT,                  ! Compute amount of free memory left
;    56          0189  1     DBG$GET_MEMORY,                     ! Get a memory block
;    57          0190  1     DBG$GET_TEMPMEM,                    ! Get a temporary memory block
```

```
;   58          0191  1     DBG$INIT_MEMORY:    NOVALUE,      ! Initialize the free memory pool
;   59          0192  1     DBG$NPARSE_ALLOCATE,              ! Parse the ALLOCATE command
;   60          0193  1     DBG$NEXECUTE_ALLOCATE,            ! Execute the ALLOCATE command
;   61          0194  1     DBG$PARSE_ALLOCATE,               ! Parse the ALLOCATE command for the
;   62          0195  1                                       !       old languages
;   63          0196  1     DBG$POP_TEMPMEM:    NOVALUE,      ! Pop the pointer to the temporary memory
;   64          0197  1                                       !       on the stack, and release that
;   65          0198  1                                       !       temporary memory blocks
;   66          0199  1     DBG$PUSH_TEMPMEM,                 ! Push the pointer to the temporary memory
;   67          0200  1                                       !       on the stack
;   68          0201  1     DBG$REL_MEMORY:     NOVALUE,      ! Release a memory block
;   69          0202  1     DBG$REL_TEMPMEM:    NOVALUE;      ! Release all temporary memory blocks
```

```
;    71          0203  1 EXTERNAL ROUTINE
;    72          0204  1     DBG$NMAKE_ARG_VECT,                ! Construct an error message vector
;    73          0205  1     DBG$NMATCH,                        ! Match a character string
;    74          0206  1     DBG$NSAVE_DECIMAL_INTEGER,         ! Parse a decimal integer
;    75          0207  1     DBG$RST_REMOVE: NOVALUE;           ! Remove the RST for a specified module
;    76          0208  1
;    77          0209  1 EXTERNAL
;    78          0210  1     DBG$GV_CONTROL: DBG$CONTROL_FLAGS, ! DEBUG control bits
;    79          0211  1     LRUM$LISTHEAD: REF LRUM$ENTRY,     ! Pointer to list head for LRUM (Least
;    80          0212  1                                        !     Recently Used Module) linked list
;    81          0213  1     RST$REF_LIST: REF VECTOR[,LONG];   ! Pointer to RST Reference List
;    82          0214  1
;    83          0215  1
;    84          0216  1 OWN
;    85          0217  1     DBG$FREE_LIST: REF FMEM$BLOCK      ! Pointer to the free-list list head
;    86          0218  1                       INITIAL(0),
;    87          0219  1     DBG$TEMP_MEMORY: INITIAL(0),       ! Pointer to the singly linked list of
;    88          0220  1                                        !     "temporary" memory blocks
;    89          0221  1     DBG$TEMPMEM_POOLID: INITIAL(0),    ! Index to the temporary memory pool
;    90          0222  1                                        !     stack
;    91          0223  1     DBG$TEMPMEM_POOLSTK: VECTOR[25],   ! Temporary memory pool stack for push
;    92          0224  1                                        !     and pop operations
;    93          0225  1     FMEM_BLOCK_LIST: INITIAL(0);       ! Pointer to a singly linked list of
;    94          0226  1                                        !     memory pool areas.
```

```
;    96          0227  1 GLOBAL ROUTINE DBG$CHECK_MEMORY: NOVALUE =
;    97          0228  1 !
;    98          0229  1 ! FUNCTION
;    99          0230  1 !       This routine checks the Debugger's memory pool for integrity.  A com-
;   100          0231  1 !       plete scan is made over the entire memory pool to check that every free
;   101          0232  1 !       and allocated block has the proper format.  A second complete scan is
;   102          0233  1 !       made over the entire memory pool free-list to verify that every free
;   103          0234  1 !       block has the proper format and that the list is intact.  A third com-
;   104          0235  1 !       plete scan is made over the "temporary block" list, and each such block
;   105          0236  1 !       is checked for validity and consistency.  If any error is detected in
;   106          0237  1 !       any of these scans, an error is signalled (Internal Memory Error) which
;   107          0238  1 !       prints the address of the bad memory block.  If the memory pool is found
;   108          0239  1 !       to be correct, the routine returns normally.
;   109          0240  1 !
;   110          0241  1 ! INPUTS
;   111          0242  1 !       NONE
;   112          0243  1 !
;   113          0244  1 ! OUTPUTS
;   114          0245  1 !       NONE
;   115          0246  1 !
;   116          0247  1
;   117          0248  2     BEGIN
;   118          0249  2
;   119          0250  2     LOCAL
;   120          0251  2         AREAPTR: REF VECTOR[,LONG],     ! Pointer to current memory pool area
;   121          0252  2         BACKPTR: REF FMEM$BLOCK,        ! Pointer to previous free-list block
;   122          0253  2         BLKADDR: REF VECTOR[,LONG],     ! Pointer to "temporary" memory block
;   123          0254  2         BLKPTR:  REF FMEM$BLOCK,        ! Pointer to the current memory block
;   124          0255  2         FREECOUNT1,                     ! Number of free blocks in memory pool
;   125          0256  2         FREECOUNT2,                     ! Number of free blocks on free-list
;   126          0257  2         LISTHEAD_FOUND,                 ! Flag indicating free-list list head
;   127          0258  2                                         !       node has been found
;   128          0259  2         NEXTBLK: REF FMEM$BLOCK,        ! Pointer to next sequential memory blk
;   129          0260  2         TEMPBLK: REF FMEM$BLOCK;        ! Pointer to "temporary" block header
;   130          0261  2
;   131          0262  2
;   132          0263  2
;   133          0264  2     ! Loop over all the memory pool areas.  These areas are linked together in a
;   134          0265  2     ! singly linked, zero-terminated list.  Check each such area for integrity.
;   135          0266  2     !
;   136          0267  2     LISTHEAD_FOUND = FALSE;
;   137          0268  2     FREECOUNT1 = 0;
;   138          0269  2     AREAPTR = .FMEM_BLOCK_LIST;
;   139          0270  2     WHILE .AREAPTR NEQ 0 DO
;   140          0271  3         BEGIN
;   141          0272  3
;   142          0273  3
;   143          0274  3         ! Loop over all the memory blocks in this area.  Check each such block
;   144          0275  3         ! for consistency.
;   145          0276  3         !
;   146          0277  3         BLKPTR = AREAPTR[2];
;   147          0278  3         IF NOT .BLKPTR[FMEM$V_PREVALLOC] THEN SIGNAL(DBG$_INTMEMERR,1,.BLKPTR);
;   148          0279  3         WHILE .BLKPTR LSSU AREAPTR[-1] + .AREAPTR[1] DO
;   149          0280  4             BEGIN
;   150          0281  4
;   151          0282  4
;   152          0283  4             ! Check the block's sentinel value.  Also get the address of the
```

```
;  153          0284  4              ! next sequential block in the memory area.
;  154          0285  4              !
;  155          0286  4              IF .BLKPTR[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL
;  156          0287  4              THEN
;  157          0288  4                  SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  158          0289  4
;  159          0290  4              NEXTBLK = .BLKPTR + 4*.BLKPTR[FMEM$V_LENGTH];
;  160          0291  4
;  161          0292  4
;  162          0293  4              ! If this is an allocated block, make sure it is marked as allocated
;  163          0294  4              ! at both ends.
;  164          0295  4              !
;  165          0296  4              IF .BLKPTR[FMEM$V_THISALLOC]
;  166          0297  4              THEN
;  167          0298  5                  BEGIN
;  168          0299  5                  IF NOT .NEXTBLK[FMEM$V_PREVALLOC]
;  169          0300  5                  THEN
;  170          0301  5                      SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  171          0302  5                  END
;  172          0303  5
;  173          0304  5
;  174          0305  5              ! If this is a free block, check that the block length and the
;  175          0306  5              ! allocation bits are consistent at both ends.
;  176          0307  5              !
;  177          0308  4              ELSE
;  178          0309  5                  BEGIN
;  179          0310  5                  IF .NEXTBLK[FMEM$L_PREVLEN] NEQ .BLKPTR[FMEM$V_LENGTH]
;  180          0311  5                  THEN
;  181          0312  5                      SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  182          0313  5
;  183          0314  5
;  184          0315  5                  ! Give special treatment to the free-list list head block.
;  185          0316  5                  !
;  186          0317  5                  IF .BLKPTR EQL .DBG$FREE_LIST
;  187          0318  5                  THEN
;  188          0319  6                      BEGIN
;  189          0320  6                      IF NOT .NEXTBLK[FMEM$V_PREVALLOC]
;  190          0321  6                      THEN
;  191          0322  6                          SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  192          0323  6
;  193          0324  6                      IF .LISTHEAD_FOUND THEN SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  194          0325  6                      LISTHEAD_FOUND = TRUE;
;  195          0326  6                      END
;  196          0327  6
;  197          0328  6
;  198          0329  6                  ! This is a free block but not the free-list list head.
;  199          0330  6                  !
;  200          0331  5                  ELSE
;  201          0332  6                      BEGIN
;  202          0333  6                      FREECOUNT1 = .FREECOUNT1 + 1;
;  203          0334  6                      IF .NEXTBLK[FMEM$V_PREVALLOC]
;  204          0335  6                      THEN
;  205          0336  6                          SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  206          0337  5                      END;
;  207          0338  5
;  208          0339  4              END;
;  209          0340  4
```

```
;  210          0341  4
;  211          0342  4            ! Go to the next sequential block in the memory area and loop.
;  212          0343  4            !
;  213          0344  4            BLKPTR = .NEXTBLK;
;  214          0345  3            END;
;  215          0346
;  216          0347
;  217          0348               ! Check the validity of the terminator block at the end of the current
;  218          0349               ! memory pool area.
;  219          0350               !
;  220          0351               IF .BLKPTR NEQ AREAPTR[-1] + .AREAPTR[1]
;  221          0352               THEN
;  222          0353                   SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  223          0354
;  224          0355               IF (.BLKPTR[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL) OR
;  225          0356                  (NOT .BLKPTR[FMEM$V_THISALLOC])                 OR
;  226          0357                  (.BLKPTR[FMEM$V_LENGTH] NEQ 1)
;  227          0358               THEN
;  228          0359                   SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  229          0360
;  230          0361
;  231          0362               ! The whole area look good.  Link to the next memory pool area and loop.
;  232          0363               !
;  233          0364               AREAPTR = .AREAPTR[0];
;  234          0365               END;
;  235          0366
;  236          0367
;  237          0368          ! Make sure the free-list list head was found in the memory pool.
;  238          0369          !
;  239          0370          IF NOT .LISTHEAD_FOUND THEN SIGNAL(DBG$_INTMEMERR, 1, .DBG$FREE_LIST);
;  240          0371
;  241          0372
;  242          0373          ! Now make a complete scan over the memory pool free-list.  Check each block
;  243          0374          ! on the list for consistency and check the integrity of the list itself.
;  244          0375          !
;  245          0376          FREECOUNT2 = 0;
;  246          0377          BACKPTR = .DBG$FREE_LIST;
;  247          0378          BLKPTR = .DBG$FREE_LIST[FMEM$L_FLINK];
;  248          0379          WHILE .BLKPTR NEQ .DBG$FREE_LIST DO
;  249          0380              BEGIN
;  250          0381
;  251          0382
;  252          0383              ! Make sure the top end of the free block is valid.
;  253          0384              !
;  254          0385              IF (.BLKPTR[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL) OR
;  255          0386                 (.BLKPTR[FMEM$V_THISALLOC])                    OR
;  256          0387                 (.BLKPTR[FMEM$L_BLINK] NEQ .BACKPTR)
;  257          0388              THEN
;  258          0389                  SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  259          0390
;  260          0391
;  261          0392              ! Make sure that the bottom end of the block is valid and consistent
;  262          0393              ! with the top end.
;  263          0394              !
;  264          0395              NEXTBLK = .BLKPTR + 4*.BLKPTR[FMEM$V_LENGTH];
;  265          0396              IF (.NEXTBLK[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL) OR
;  266          0397                 (.NEXTBLK[FMEM$V_PREVALLOC])                    OR
```

```
;  267          0398  4          (.NEXTBLK[FMEM$L_PREVLEN] NEQ .BLKPTR[FMEM$V_LENGTH])
;  268          0399  4      THEN
;  269          0400  4          SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  270          0401  4
;  271          0402  4
;  272          0403  4      ! This block looks good.  Link to the next block on the list and loop.
;  273          0404  4      !
;  274          0405  4      FREECOUNT2 = .FREECOUNT2 + 1;
;  275          0406  4      BACKPTR = .BLKPTR;
;  276          0407  4      BLKPTR = .BLKPTR[FMEM$L_FLINK];
;  277          0408  4      END;
;  278          0409  3
;  279          0410  3
;  280          0411  3  ! We are back at the free-list list head.  Make sure its backward link
;  281          0412  3  ! points to the last block we inspected.  Also make sure that the number
;  282          0413  3  ! of free blocks came out the same in the memory-area and free-list scans.
;  283          0414  3  !
;  284          0415  3  IF (.BLKPTR[FMEM$L_BLINK] NEQ .BACKPTR) OR
;  285          0416  3     (.FREECOUNT1 NEQ .FREECOUNT2)
;  286          0417  3  THEN
;  287          0418  3      SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  288          0419  3
;  289          0420  3
;  290          0421  3  ! Check the consistency of the "temporary" memory block chain.  Loop through
;  291          0422  3  ! the DBG$TEMP_MEMORY list (which is singly linked) and check each block.
;  292          0423  3  !
;  293          0424  3  BLKADDR = .DBG$TEMP_MEMORY;
;  294          0425  3  WHILE .BLKADDR NEQ 0 DO
;  295          0426  3      BEGIN
;  296          0427  4      TEMPBLK = BLKADDR[1];
;  297          0428  4      BLKPTR = BLKADDR[-1];
;  298          0429  4
;  299          0430  4
;  300          0431  4      ! Make sure the temporary block header looks correct.
;  301          0432  4      !
;  302          0433  4      IF (.TEMPBLK[FMEM$B_SENTINEL] NEQ FMEM$K_TEMPSENT) OR
;  303          0434  4         (NOT .TEMPBLK[FMEM$V_THISALLOC])                 OR
;  304          0435  4         (NOT .TEMPBLK[FMEM$V_PREVALLOC])
;  305          0436  4      THEN
;  306          0437  4          SIGNAL(DBG$_INTMEMERR, 1, .TEMPBLK);
;  307          0438  4
;  308          0439  4
;  309          0440  4      ! Make sure the memory pool block header before that looks correct and
;  310          0441  4      ! is consistent with the temporary block header.
;  311          0442  4      !
;  312          0443  4      IF (.BLKPTR[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL)           OR
;  313          0444  4         (NOT .BLKPTR[FMEM$V_THISALLOC])                      OR
;  314          0445  4         (.BLKPTR[FMEM$V_LENGTH] LSS .TEMPBLK[FMEM$V_LENGTH] + 2) OR
;  315          0446  4         (.BLKPTR[FMEM$V_LENGTH] GTR .TEMPBLK[FMEM$V_LENGTH] + 5)
;  316          0447  4      THEN
;  317          0448  4          SIGNAL(DBG$_INTMEMERR, 1, .TEMPBLK);
;  318          0449  4
;  319          0450  4
;  320          0451  4      ! Make sure the bottom end of the memory block looks allright too.
;  321          0452  4      !
;  322          0453  4      NEXTBLK = .BLKPTR + 4*.BLKPTR[FMEM$V_LENGTH];
;  323          0454  4      IF (.NEXTBLK[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL) OR
```

```
;  324          0455  4          (NOT .NEXTBLK[FMEM$V_PREVALLOC])
;  325          0456  3      THEN
;  326          0457  3          SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  327          0458  3
;  328          0459  3
;  329          0460  3      ! This block looks good. Link to the next block and loop.
;  330          0461  3      !
;  331          0462  3      BLKADDR = .BLKADDR[0];
;  332          0463  2      END;
;  333          0464  2
;  334          0465  2
;  335          0466  2  ! The memory pool passes all tests.  Return successfully to the caller.
;  336          0467  2  !
;  337          0468  2  RETURN;
;  338          0469  2
;  339          0470  1  END;

                                                .TITLE  GETMEMORY
                                                .IDENT  \V04-000\

                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                    00000000  00000 DBG$FREE_LIST:
                                                .LONG   0                                       ;
                                    00000000  00004 DBG$TEMP_MEMORY:
                                                .LONG   0                                       ;
                                    00000000  00008 DBG$TEMPMEM_POOLID:
                                                .LONG   0                                       ;
                                              0000C DBG$TEMPMEM_POOLSTK:
                                                .BLKB   100
                                    00000000  00070 FMEM_BLOCK_LIST:
                                                .LONG   0                                       ;

                                                .EXTRN  DBG$NMAKE_ARG_VECT
                                                .EXTRN  DBG$NMATCH, DBG$NSAVE_DECIMAL_INTEGER
                                                .EXTRN  DBG$RST_REMOVE, DBG$GV_CONTROL
                                                .EXTRN  LRUM$LISTHEAD, RST$REF_LIST

                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                         03FC 00000             .ENTRY  DBG$CHECK_MEMORY, Save R2,R3,R4,R5,R6,R7,-      ; 0227
                                                                        R8,R9                   ;
                                59 00000000'   EF   9E 00002            MOVAB   DBG$FREE_LIST, R9                       ;
                                58 00000000G   00   9E 00009            MOVAB   LIB$SIGNAL, R8                          ;
                                               56   7C 00010            CLRQ    LISTHEAD_FOUND                          ; 0267
                                53        70   A9   D0 00012            MOVL    FMEM_BLOCK_LIST, AREAPTR                ; 0269
                                               03   12 00016 1$:         BNEQ    2$                                      ; 0270
                                            00D4   31 00018             BRW     16$                                     ;
                                52        08   A3   9E 0001B 2$:        MOVAB   8(R3), BLKPTR                           ; 0277
                          0D    62             17   E0 0001F            BBS     #23, (BLKPTR), 3$                       ; 0278
                                               52   DD 00023            PUSHL   BLKPTR                                  ;
                                               01   DD 00025            PUSHL   #1                                      ;
                                   00028AF4    8F   DD 00027            PUSHL   #166644                                 ;
                                68             03   FB 0002D            CALLS   #3, LIB$SIGNAL                          ;
                          54    53        04   A3   C1 00030 3$:        ADDL3   4(AREAPTR), AREAPTR, R4                 ; 0279
                                54             04   C2 00035            SUBL2   #4, R4                                  ;
```

```
                        54        52  D1 00038          CMPL    BLKPTR, R4                                      ;
                                  7E  1E 0003B          BGEQU   12$                                             ;
                    B2  8F    03  A2  91 0003D          CMPB    3(BLKPTR), #178                                 ; 0286
                                  0D  13 00042          BEQL    4$                                              ;
                                  52  DD 00044          PUSHL   BLKPTR                                          ; 0288
                                  01  DD 00046          PUSHL   #1                                              ;
                        00028AF4  8F  DD 00048          PUSHL   #166644                                         ;
                        68        03  FB 0004E          CALLS   #3, LIB$SIGNAL                                  ;
              50        62        16  00  EF 00051 4$:  EXTZV   #0, #22, (BLKPTR), R0                          ; 0290
                        55      6240  DE 00056          MOVAL   (BLKPTR)[R0], NEXTBLK                           ;
                    06  62        16  E1 0005A          BBC     #22, (BLKPTR), 5$                               ; 0296
                    53  65        17  E0 0005E          BBS     #23, (NEXTBLK), 11$                             ; 0299
                                  44  11 00062          BRB     10$                                             ; 0301
                        50    FC  A5  D1 00064 5$:      CMPL    -4(NEXTBLK), R0                                 ; 0310
                                  0D  13 00068          BEQL    6$                                              ;
                                  52  DD 0006A          PUSHL   BLKPTR                                          ; 0312
                                  01  DD 0006C          PUSHL   #1                                              ;
                        00028AF4  8F  DD 0006E          PUSHL   #166644                                         ;
                        68        03  FB 00074          CALLS   #3, LIB$SIGNAL                                  ;
                        69        52  D1 00077 6$:      CMPL    BLKPTR, DBG$FREE_LIST                           ; 0317
                                  26  12 0007A          BNEQ    9$                                              ;
                    0D  65        17  E0 0007C          BBS     #23, (NEXTBLK), 7$                              ; 0320
                                  52  DD 00080          PUSHL   BLKPTR                                          ; 0322
                                  01  DD 00082          PUSHL   #1                                              ;
                        00028AF4  8F  DD 00084          PUSHL   #166644                                         ;
                        68        03  FB 0008A          CALLS   #3, LIB$SIGNAL                                  ;
                        0D        56  E9 0008D 7$:      BLBC    LISTHEAD_FOUND, 8$                              ; 0324
                                  52  DD 00090          PUSHL   BLKPTR                                          ;
                                  01  DD 00092          PUSHL   #1                                              ;
                        00028AF4  8F  DD 00094          PUSHL   #166644                                         ;
                        68        03  FB 0009A          CALLS   #3, LIB$SIGNAL                                  ;
                        56        01  D0 0009D 8$:      MOVL    #1, LISTHEAD_FOUND                              ; 0325
                                  13  11 000A0          BRB     11$                                             ; 0317
                                  57  D6 000A2 9$:      INCL    FREECOUNT1                                      ; 0333
                    0D  65        17  E1 000A4          BBC     #23, (NEXTBLK), 11$                             ; 0334
                                  52  DD 000A8 10$:      PUSHL   BLKPTR                                          ; 0336
                                  01  DD 000AA          PUSHL   #1                                              ;
                        00028AF4  8F  DD 000AC          PUSHL   #166644                                         ;
                        68        03  FB 000B2          CALLS   #3, LIB$SIGNAL                                  ;
                        52        55  D0 000B5 11$:      MOVL    NEXTBLK, BLKPTR                                 ; 0344
                                FF75  31 000B8          BRW     3$                                              ; 0279
                                  0D  13 000BB 12$:      BEQL    13$                                             ; 0351
                                  52  DD 000BD          PUSHL   BLKPTR                                          ; 0353
                                  01  DD 000BF          PUSHL   #1                                              ;
                        00028AF4  8F  DD 000C1          PUSHL   #166644                                         ;
                        68        03  FB 000C7          CALLS   #3, LIB$SIGNAL                                  ;
                    B2  8F    03  A2  91 000CA 13$:      CMPB    3(BLKPTR), #178                                 ; 0355
                                  0B  12 000CF          BNEQ    14$                                             ;
                    07  62        16  E1 000D1          BBC     #22, (BLKPTR), 14$                              ; 0356
              01    62        16  00  ED 000D5          CMPZV   #0, #22, (BLKPTR), #1                           ; 0357
                                  0D  13 000DA          BEQL    15$                                             ;
                                  52  DD 000DC 14$:      PUSHL   BLKPTR                                          ; 0359
                                  01  DD 000DE          PUSHL   #1                                              ;
                        00028AF4  8F  DD 000E0          PUSHL   #166644                                         ;
                        68        03  FB 000E6          CALLS   #3, LIB$SIGNAL                                  ;
                        53        63  D0 000E9 15$:      MOVL    (AREAPTR), AREAPTR                              ; 0364
                                FF27  31 000EC          BRW     1$                                              ; 0270
```

```
                                0D              56  E8 000EF 16$:    BLBS    LISTHEAD_FOUND, 17$                     ; 0370
                                                69  DD 000F2       PUSHL   DBG$FREE_LIST                           ;
                                                01  DD 000F4       PUSHL   #1                                      ;
                                    00028AF4    8F  DD 000F6       PUSHL   #166644                                 ;
                                68              03  FB 000FC       CALLS   #3, LIB$SIGNAL                          ;
                                                54  D4 000FF 17$:   CLRL    FREECOUNT2                              ; 0376
                                50              69  D0 00101       MOVL    DBG$FREE_LIST, R0                       ; 0377
                                53              50  D0 00104       MOVL    R0, BACKPTR                             ;
                                52          04  A0  D0 00107       MOVL    4(R0), BLKPTR                           ; 0378
                                69              52  D1 0010B 18$:   CMPL    BLKPTR, DBG$FREE_LIST                   ; 0379
                                                50  13 0010E       BEQL    23$                                     ;
                            B2  8F          03  A2  91 00110       CMPB    3(BLKPTR), #178                         ; 0385
                                                0A  12 00115       BNEQ    19$                                     ;
                        06                  62  16  E0 00117       BBS     #22, (BLKPTR), 19$                      ; 0386
                                53          08  A2  D1 0011B       CMPL    8(BLKPTR), BACKPTR                      ; 0387
                                                0D  13 0011F       BEQL    20$                                     ;
                                                52  DD 00121 19$:   PUSHL   BLKPTR                                 ; 0389
                                                01  DD 00123       PUSHL   #1                                      ;
                                    00028AF4    8F  DD 00125       PUSHL   #166644                                 ;
                                68              03  FB 0012B       CALLS   #3, LIB$SIGNAL                          ;
                    50          62              00  EF 0012E 20$:   EXTZV   #0, #22, (BLKPTR), R0                   ; 0395
                                55          6240  DE 00133       MOVAL   (BLKPTR)[R0], NEXTBLK                   ;
                            B2  8F          03  A5  91 00137       CMPB    3(NEXTBLK), #178                        ; 0396
                                                0A  12 0013C       BNEQ    21$                                     ;
                        06                  65  17  E0 0013E       BBS     #23, (NEXTBLK), 21$                     ; 0397
                                50          FC  A5  D1 00142       CMPL    -4(NEXTBLK), R0                         ; 0398
                                                0D  13 00146       BEQL    22$                                     ;
                                                52  DD 00148 21$:   PUSHL   BLKPTR                                 ; 0400
                                                01  DD 0014A       PUSHL   #1                                      ;
                                    00028AF4    8F  DD 0014C       PUSHL   #166644                                 ;
                                68              03  FB 00152       CALLS   #3, LIB$SIGNAL                          ;
                                                54  D6 00155 22$:   INCL    FREECOUNT2                              ; 0405
                                53              52  D0 00157       MOVL    BLKPTR, BACKPTR                         ; 0406
                                52          04  A2  D0 0015A       MOVL    4(BLKPTR), BLKPTR                       ; 0407
                                                AB  11 0015E       BRB     18$                                     ; 0379
                                53          08  A2  D1 00160 23$:   CMPL    8(BLKPTR), BACKPTR                      ; 0415
                                                05  12 00164       BNEQ    24$                                     ;
                                54              57  D1 00166       CMPL    FREECOUNT1, FREECOUNT2                  ; 0416
                                                0D  13 00169       BEQL    25$                                     ;
                                                52  DD 0016B 24$:   PUSHL   BLKPTR                                 ; 0418
                                                01  DD 0016D       PUSHL   #1                                      ;
                                    00028AF4    8F  DD 0016F       PUSHL   #166644                                 ;
                                68              03  FB 00175       CALLS   #3, LIB$SIGNAL                          ;
                                54          04  A9  D0 00178 25$:   MOVL    DBG$TEMP_MEMORY, BLKADDR                ; 0424
                                                01  12 0017C 26$:   BNEQ    27$                                     ; 0425
                                                04  0017E          RET                                             ;
                                53          04  A4  9E 0017F 27$:   MOVAB   4(R4), TEMPBLK                         ; 0427
                                52          FC  A4  9E 00183       MOVAB   -4(R4), BLKPTR                          ; 0428
                            B4  8F          03  A3  91 00187       CMPB    3(TEMPBLK), #180                        ; 0433
                                                08  12 0018C       BNEQ    28$                                     ;
                    04                      63  16  E1 0018E       BBC     #22, (TEMPBLK), 28$                     ; 0434
                    0D                      63  17  E0 00192       BBS     #23, (TEMPBLK), 29$                     ; 0435
                                                53  DD 00196 28$:   PUSHL   TEMPBLK                                 ; 0437
                                                01  DD 00198       PUSHL   #1                                      ;
                                    00028AF4    8F  DD 0019A       PUSHL   #166644                                 ;
                                68              03  FB 001A0       CALLS   #3, LIB$SIGNAL                          ;
                            B2  8F          03  A2  91 001A3 29$:   CMPB    3(BLKPTR), #178                         ; 0443
```

```
                                        22  12 001A8         BNEQ    30$                              ;
                1E          62          16  E1 001AA         BBC     #22, (BLKPTR), 30$               ; 0444
      50        63          16          00  EF 001AE         EXTZV   #0, #22, (TEMPBLK), R0           ; 0445
                            50          02  C0 001B3         ADDL2   #2, R0                           ;
      50        62          16          00  ED 001B6         CMPZV   #0, #22, (BLKPTR), R0            ;
                                        0F  19 001BB         BLSS    30$                              ;
      50        63          16          00  EF 001BD         EXTZV   #0, #22, (TEMPBLK), R0           ; 0446
                            50          05  C0 001C2         ADDL2   #5, R0                           ;
      50        62          16          00  ED 001C5         CMPZV   #0, #22, (BLKPTR), R0            ;
                                        0D  15 001CA         BLEQ    31$                              ;
                                        53  DD 001CC 30$:    PUSHL   TEMPBLK                          ; 0448
                                        01  DD 001CE         PUSHL   #1                               ;
                            00028AF4    8F  DD 001D0         PUSHL   #166644                          ;
                            68          03  FB 001D6         CALLS   #3, LIB$SIGNAL                   ;
      50        62          16          00  EF 001D9 31$:    EXTZV   #0, #22, (BLKPTR), R0            ; 0453
                            55        6240  DE 001DE         MOVAL   (BLKPTR)[R0], NEXTBLK            ;
                    B2      8F     03   A5  91 001E2         CMPB    3(NEXTBLK), #178                 ; 0454
                                        04  12 001E7         BNEQ    32$                              ;
                0D          65          17  E0 001E9         BBS     #23, (NEXTBLK), 33$              ; 0455
                                        52  DD 001ED 32$:    PUSHL   BLKPTR                           ; 0457
                                        01  DD 001EF         PUSHL   #1                               ;
                            00028AF4    8F  DD 001F1         PUSHL   #166644                          ;
                            68          03  FB 001F7         CALLS   #3, LIB$SIGNAL                   ;
                            54          64  D0 001FA 33$:    MOVL    (BLKADDR), BLKADDR               ; 0462
                                      FF7C  31 001FD         BRW     26$                              ; 0425
                                        04     00200         RET                                      ; 0470

; Routine Size:  513 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  341     0471  1 GLOBAL ROUTINE DBG$COPY_MEMORY(SOURCE) =
;  342     0472  1 !
;  343     0473  1 ! FUNCTION
;  344     0474  1 !       This routine creates a new block of memory and copies the contents of a
;  345     0475  1 !       specified block to the new block.  The new block is made large enough to
;  346     0476  1 !       hold the entire contents of the specified block--the length of the spec-
;  347     0477  1 !       ified block is gotten from the block's control longword (FMEM$V_LENGTH).
;  348     0478  1 !       Note that the new block is a "permanent" memory block--it is not put on
;  349     0479  1 !       the temporary block list.
;  350     0480  1 !
;  351     0481  1 ! INPUTS
;  352     0482  1 !       SOURCE  - A pointer to the memory block to be copied.  This may be
;  353     0483  1 !                 either a permanent or a temporary block.
;  354     0484  1 !
;  355     0485  1 ! OUTPUTS
;  356     0486  1 !       A new block is allocated and the contents of the SOURCE block is copied
;  357     0487  1 !                 to the new block.  The address of the new block is returned as
;  358     0488  1 !                 the routine's value.
;  359     0489  1 !
;  360     0490  1
;  361     0491  2     BEGIN
;  362     0492  2
;  363     0493  2     MAP
;  364     0494  2         SOURCE: REF FMEM$BLOCK;          ! Pointer to allocated portion of the
;  365     0495  2                                          !      memory block to be copied
;  366     0496  2
;  367     0497  2     LOCAL
;  368     0498  2         LENGTH,                          ! The length of the source and target
;  369     0499  2                                          !      blocks in longwords
;  370     0500  2         SRCBLK: REF FMEM$BLOCK,          ! Pointer to source block control info
;  371     0501  2         TARGET;                          ! The address of the new memory block
;  372     0502  2
;  373     0503  2
;  374     0504  2
;  375     0505  2     ! Pick up the address of the source block's control information and check
;  376     0506  2     ! that it is a valid memory block (permanent or temporary).
;  377     0507  2     !
;  378     0508  2     SRCBLK = SOURCE[FMEM$A_HEADER];
;  379     0509  2     IF .SRCBLK[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL AND
;  380     0510  2        .SRCBLK[FMEM$B_SENTINEL] NEQ FMEM$K_TEMPSENT
;  381     0511  2     THEN
;  382     0512  2         SIGNAL(DBG$_INTMEMERR, 1, .SRCBLK);
;  383     0513  2
;  384     0514  2
;  385     0515  2     ! Get the length of the source block, allocate the target block, copy the
;  386     0516  2     ! the contents from the source to the target, and return the target address.
;  387     0517  2     !
;  388     0518  2     LENGTH = .SRCBLK[FMEM$V_LENGTH] - 1;
;  389     0519  2     TARGET = DBG$GET_MEMORY(.LENGTH);
;  390     0520  2     CH$MOVE(4*.LENGTH, .SOURCE, .TARGET);
;  391     0521  2     RETURN .TARGET;
;  392     0522  2
;  393     0523  1     END;
```

```
                                                 007C 00000          .ENTRY  DBG$COPY_MEMORY, Save R2,R3,R4,R5,R6    ; 0471
                52          04    AC                 04   C3 00002  SUBL3   #4, SOURCE, SRCBLK                    ; 0508
                            B2    8F         03      A2   91 00007  CMPB    3(SRCBLK), #178                      ; 0509
                                                     18   13 0000C  BEQL    1$                                   ;
                            B4    8F         03      A2   91 0000E  CMPB    3(SRCBLK), #180                      ; 0510
                                                     11   13 00013  BEQL    1$                                   ;
                                                     52   DD 00015  PUSHL   SRCBLK                               ; 0512
                                                     01   DD 00017  PUSHL   #1                                   ;
                                          00028AF4   8F   DD 00019  PUSHL   #166644                               ;
                      00000000G   00                 03   FB 0001F  CALLS   #3, LIB$SIGNAL                      ;
        52          62          16                 00   EF 00026  1$:  EXTZV   #0, #22, (SRCBLK), LENGTH         ; 0518
                                                     72   9F 0002B  PUSHAB  -(LENGTH)                            ; 0519
                      0000V     CF                 01   FB 0002D  CALLS   #1, DBG$GET_MEMORY                    ;
                                56                 50   D0 00032  MOVL    R0, TARGET                           ;
                                52                 04   C4 00035  MULL2   #4, R2                               ; 0520
                66          04    BC                 52   28 00038  MOVC3   R2, @SOURCE, (TARGET)               ;
                                50                 56   D0 0003D  MOVL    TARGET, R0                           ; 0521
                                                        04 00040  RET                                           ; 0523

; Routine Size:  65 bytes,    Routine Base:  DBG$CODE + 0201
```

```
;  395          0524  1 GLOBAL ROUTINE DBG$COPY_TEMPMEM(SOURCE) =
;  396          0525  1 !
;  397          0526  1 ! FUNCTION
;  398          0527  1 !       This routine creates a new temporary block of memory and copies the con-
;  399          0528  1 !       tents of a specified block to that new block.  The new temporary block
;  400          0529  1 !       is made large enough to hold the entire contents of the specified block
;  401          0530  1 !       --the block's length is gotten from FMEM$V_LENGTH.  Since the new block
;  402          0531  1 !       is a "temporary" block, it disappears at the end of the current command.
;  403          0532  1 !
;  404          0533  1 ! INPUTS
;  405          0534  1 !       SOURCE  - A pointer to the memory block to be copied.  This may be
;  406          0535  1 !                 either a permanent or a temporary block.
;  407          0536  1 !
;  408          0537  1 ! OUTPUTS
;  409          0538  1 !       A temporary block is allocated and the contents of the SOURCE block is
;  410          0539  1 !                 copied to the temporary block.  The address of the temporary
;  411          0540  1 !                 block is returned as the routine's value.
;  412          0541  1 !
;  413          0542  1
;  414          0543  2     BEGIN
;  415          0544  2
;  416          0545  2     MAP
;  417          0546  2         SOURCE: REF FMEM$BLOCK;          ! Pointer to allocated portion of the
;  418          0547  2                                          !      memory block to be copied
;  419          0548  2
;  420          0549  2     LOCAL
;  421          0550  2         LENGTH,                          ! The length of the source and target
;  422          0551  2                                          !      blocks in longwords
;  423          0552  2         SRCBLK: REF FMEM$BLOCK,          ! Pointer to source block control info
;  424          0553  2         TARGET;                          ! The address of the new memory block
;  425          0554  2
;  426          0555  2
;  427          0556  2
;  428          0557  2     ! Pick up the address of the source block's control information and check
;  429          0558  2     ! that it is a valid memory block (permanent or temporary).
;  430          0559  2     !
;  431          0560  2     SRCBLK = SOURCE[FMEM$A_HEADER];
;  432          0561  2     IF .SRCBLK[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL AND
;  433          0562  2        .SRCBLK[FMEM$B_SENTINEL] NEQ FMEM$K_TEMPSENT
;  434          0563  2     THEN
;  435          0564  2         SIGNAL(DBG$_INTMEMERR, 1, .SRCBLK);
;  436          0565  2
;  437          0566  2
;  438          0567  2     ! Get the length of the source block, allocate the target block, copy the
;  439          0568  2     ! the contents from the source to the target, and return the target address.
;  440          0569  2     !
;  441          0570  2     LENGTH = .SRCBLK[FMEM$V_LENGTH] - 1;
;  442          0571  2     TARGET = DBG$GET_TEMPMEM(.LENGTH);
;  443          0572  2     CH$MOVE(4*.LENGTH, .SOURCE, .TARGET);
;  444          0573  2     RETURN .TARGET;
;  445          0574  2
;  446          0575  1     END;
```

```
                                                  007C 00000        .ENTRY  DBG$COPY_TEMPMEM, Save R2,R3,R4,R5,R6     ; 0524
                 52        04   AC           04   C3 00002          SUBL3   #4, SOURCE, SRCBLK                        ; 0560
                           B2   8F      03   A2   91 00007          CMPB    3(SRCBLK), #178                           ; 0561
                                             18   13 0000C          BEQL    1$                                        ;
                           B4   8F      03   A2   91 0000E          CMPB    3(SRCBLK), #180                           ; 0562
                                             11   13 00013          BEQL    1$                                        ;
                                             52   DD 00015          PUSHL   SRCBLK                                    ; 0564
                                             01   DD 00017          PUSHL   #1                                        ;
                                 00028AF4    8F   DD 00019          PUSHL   #166644                                   ;
                      00000000G  00          03   FB 0001F          CALLS   #3, LIB$SIGNAL                            ;
     52          62              16          00   EF 00026 1$:      EXTZV   #0, #22, (SRCBLK), LENGTH                 ; 0570
                                             72   9F 0002B          PUSHAB  -(LENGTH)                                 ; 0571
                          0000V  CF          01   FB 0002D          CALLS   #1, DBG$GET_TEMPMEM                       ;
                                 56          50   D0 00032          MOVL    R0, TARGET                                ;
                                 52          04   C4 00035          MULL2   #4, R2                                    ; 0572
                 66        04    BC          52   28 00038          MOVC3   R2, @SOURCE, (TARGET)                     ;
                                 50          56   D0 0003D          MOVL    TARGET, R0                                ; 0573
                                                  04 00040          RET                                               ; 0575

; Routine Size:  65 bytes,    Routine Base:  DBG$CODE + 0242
```

```
;  448          0576  1 GLOBAL ROUTINE DBG$EXPAND_MEMORY(LENGTH) =
;  449          0577  1 !
;  450          0578  1 ! FUNCTION
;  451          0579  1 !       This routine expands the free memory pool.  To do so, it calls the
;  452          0580  1 !       $EXPREG system service to get a new memory pool area at the end of P0
;  453          0581  1 !       space.  This area is initialized to contain one big free block and a
;  454          0582  1 !       one longword terminator block, after which the free block is linked
;  455          0583  1 !       into the memory pool free-list.
;  456          0584  1 !
;  457          0585  1 !       This routine is called during the Debugger's initialization phase and
;  458          0586  1 !       in response to the ALLOCATE and SET MODULE/ALLOCATE commands.  It is
;  459          0587  1 !       never called automatically after DEBUG has given the user program
;  460          0588  1 !       control unless the user has explicitly requested it (via the ALLOCATE
;  461          0589  1 !       command or qualifier).  The reason is that memory expansions after the
;  462          0590  1 !       user program has started can cause checkerboarding of DEBUG's and the
;  463          0591  1 !       user program's memory, which may affect the user program's execution in
;  464          0592  1 !       unpredictable ways.  Such checkerboarding changes address relationships
;  465          0593  1 !       in the user program, which can change program behavior, and it makes it
;  466          0594  1 !       much more likely that the user program will overwrite part of DEBUG's
;  467          0595  1 !       memory pool.  These possibilities are particularly undesirable since
;  468          0596  1 !       the user program is being debugged and can be presumed to have errors,
;  469          0597  1 !       possibly including random addressing errors.
;  470          0598  1 !
;  471          0599  1 ! INPUTS
;  472          0600  1 !       LENGTH  - The number of longwords to be added to the free memory pool.
;  473          0601  1 !                 This must be at least 4 and at most 3FFFFF hex.
;  474          0602  1 !
;  475          0603  1 ! OUTPUTS
;  476          0604  1 !       The new memory area is acquired and added to the memory pool free-list.
;  477          0605  1 !       One of these two values is returned:
;  478          0606  1 !
;  479          0607  1 !               STS$K_SUCCESS -- All went well and the memory is available.
;  480          0608  1 !               STS$K_SEVERE  -- The requested memory could not be gotten
;  481          0609  1 !                                from $EXPREG.  The memory pool was thus
;  482          0610  1 !                                not expanded.
;  483          0611  1 !
;  484          0612  1
;  485          0613  2     BEGIN
;  486          0614  2
;  487          0615  2     LOCAL
;  488          0616  2         BACKPTR: REF FMEM$BLOCK,        ! Pointer to previous free-list block
;  489          0617  2         ENDPTR:  REF FMEM$BLOCK,        ! Pointer to end of the memory pool area
;  490          0618  2         FORWPTR: REF FMEM$BLOCK,        ! Pointer to the next free-list block
;  491          0619  2         FREEBLK: REF FMEM$BLOCK,        ! Pointer to the one big free block
;  492          0620  2         MEMBOUNDS: VECTOR[2, LONG],     ! Start and end addresses of new area
;  493          0621  2         MEMVECTOR: REF VECTOR[,LONG],   ! Pointer to acquired memory area
;  494          0622  2         NUMLONGS,                       ! Number of longwords actually gotten
;  495          0623  2                                         !       from $EXPREG for free block
;  496          0624  2         NUMPAGES,                       ! Number of pages to get from $EXPREG
;  497          0625  2         STATUS: BLOCK[1, LONG];         ! Status code returned by $EXPREG
;  498          0626  2
;  499          0627  2
;  500          0628  2
;  501          0629  2     ! Check that the LENGTH parameter is in the valid range.
;  502          0630  2     !
;  503          0631  2     IF .LENGTH LSS 4
;  504          0632  2     THEN
```

```
;  505          0633  2          $DBG_ERROR('GETMEMORY\DBG$EXPAND_MEMORY');
;  506          0634  2      IF .LENGTH GTR %X'3FFFFF' THEN RETURN STS$K_SEVERE;
;  507          0635  2
;  508          0636  2
;  509          0637  2      ! Get the desired amount of memory from $EXPREG.  Note that we request
;  510          0638  2      ! three extra longwords for control information and the terminator block.
;  511          0639  2      ! We also round up to the nearest page boundary.
;  512          0640  2      !
;  513          0641  2      NUMPAGES = (.LENGTH + 3 + 127)/128;
;  514          0642  2      STATUS = $EXPREG(PAGCNT=.NUMPAGES, RETADR=MEMBOUNDS);
;  515          0643  2      IF NOT .STATUS
;  516          0644  2      THEN
;  517          0645  3          BEGIN
;  518          0646  3          SIGNAL(DBG$_INSVIRMEM, .STATUS);
;  519          0647  3          RETURN STS$K_SEVERE;
;  520          0648  2          END;
;  521          0649  2
;  522          0650  2      MEMVECTOR = .MEMBOUNDS[0];
;  523          0651  2      NUMLONGS = (.MEMBOUNDS[1] - .MEMBOUNDS[0] + 1)/%UPVAL - 3;
;  524          0652  2
;  525          0653  2
;  526          0654  2      ! We got the new memory pool area.  Link this area into the singly linked
;  527          0655  2      ! list of memory pool areas and remember its byte length.  Also set up the
;  528          0656  2      ! pointers to the one big free block and the terminator block.
;  529          0657  2      !
;  530          0658  2      MEMVECTOR[0] = .FMEM_BLOCK_LIST;
;  531          0659  2      MEMVECTOR[1] = (.NUMLONGS + 3)*%UPVAL;
;  532          0660  2      FMEM_BLOCK_LIST = MEMVECTOR[0];
;  533          0661  2      FREEBLK = MEMVECTOR[2];
;  534          0662  2      ENDPTR = MEMVECTOR[.NUMLONGS + 2];
;  535          0663  2
;  536          0664  2
;  537          0665  2      ! Set up the one big free block in the memory area.  Note that we mark the
;  538          0666  2      ! previous "block" as being allocated--this prevents any attempt to coalesce
;  539          0667  2      ! free blocks off the top of the memory area.  Link the free block into the
;  540          0668  2      ! memory pool free-list.
;  541          0669  2      !
;  542          0670  2      FREEBLK[FMEM$V_LENGTH] = .NUMLONGS;
;  543          0671  2      FREEBLK[FMEM$V_THISALLOC] = FALSE;
;  544          0672  2      FREEBLK[FMEM$V_PREVALLOC] = TRUE;
;  545          0673  2      FREEBLK[FMEM$B_SENTINEL] = FMEM$K_SENTINEL;
;  546          0674  2      ENDPTR[FMEM$L_PREVLEN] = .FREEBLK[FMEM$V_LENGTH];
;  547          0675  2      FORWPTR = .DBG$FREE_LIST[FMEM$L_FLINK];
;  548          0676  2      BACKPTR = .DBG$FREE_LIST;
;  549          0677  2      FREEBLK[FMEM$L_FLINK] = .FORWPTR;
;  550          0678  2      FREEBLK[FMEM$L_BLINK] = .BACKPTR;
;  551          0679  2      FORWPTR[FMEM$L_BLINK] = .FREEBLK;
;  552          0680  2      BACKPTR[FMEM$L_FLINK] = .FREEBLK;
;  553          0681  2
;  554          0682  2
;  555          0683  2      ! Finally build the terminator block.  This "block" is marked as allocated
;  556          0684  2      ! so that we will not coalesce free blocks off the end of the memory area.
;  557          0685  2      ! Then return successful status.
;  558          0686  2      !
;  559          0687  2      ENDPTR[FMEM$V_LENGTH] = 1;
;  560          0688  2      ENDPTR[FMEM$V_THISALLOC] = TRUE;
;  561          0689  2      ENDPTR[FMEM$V_PREVALLOC] = FALSE;
```

```
;  562          0690  2     ENDPTR[FMEM$B_SENTINEL] = FMEM$K_SENTINEL;
;  563          0691  2     RETURN STS$K_SUCCESS;
;  564          0692  2
;  565          0693  1     END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

24 47 42 44 5C 59 52 4F 4D 45 4D 54 45 47 1B  00000 P.AAA:  .ASCII  <27>\GETMEMORY\<92>\DBG$EXPAND_MEMORY\      ;
      59 52 4F 4D 45 4D 5F 44 4E 41 50 58 45  0000F                                                           ;

                                                        .EXTRN  SYS$EXPREG

                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                          003C 00000    .ENTRY  DBG$EXPAND_MEMORY, Save R2,R3,R4,R5           ; 0576
                          55 00000000G    00   9E 00002 MOVAB   LIB$SIGNAL, R5                               ;
                          54 00000000'    EF   9E 00009 MOVAB   FMEM_BLOCK_LIST, R4                          ;
                          5E              08   C2 00010 SUBL2   #8, SP                                      ;
                          04        04    AC   D1 00013 CMPL    LENGTH, #4                                  ; 0631
                                          11   18 00017 BGEQ    1$                                          ;
                             00000000'    EF   9F 00019 PUSHAB  P.AAA                                       ; 0633
                                          01   DD 0001F PUSHL   #1                                          ;
                             00028362     8F   DD 00021 PUSHL   #164706                                     ;
                          65              03   FB 00027 CALLS   #3, LIB$SIGNAL                            ;
                 003FFFFF 8F        04    AC   D1 0002A 1$:     CMPL    LENGTH, #4194303                    ; 0634
                                          2C   14 00032 BGTR    2$                                          ;
              50        04    AC 00000082 8F   C1 00034 ADDL3   #130, LENGTH, R0                            ; 0641
                          50 00000080     8F   C6 0003D DIVL2   #128, NUMPAGES                              ;
                          7E              7C   00044 CLRQ    -(SP)                                       ; 0642
                                    08    AE   9F 00046 PUSHAB  MEMBOUNDS                                   ;
                                          50   DD 00049 PUSHL   NUMPAGES                                   ;
                 00000000G 00             04   FB 0004B CALLS   #4, SYS$EXPREG                             ;
                          0F              50   E8 00052 BLBS    STATUS, 3$                                  ; 0643
                                          50   DD 00055 PUSHL   STATUS                                     ; 0646
                             00028093     8F   DD 00057 PUSHL   #163987                                     ;
                          65              02   FB 0005D CALLS   #2, LIB$SIGNAL                            ;
                          50              04   D0 00060 2$:     MOVL    #4, R0                              ; 0647
                                          04   00063 RET                                                    ;
                          50              6E   D0 00064 3$:     MOVL    MEMBOUNDS, MEMVECTOR                ; 0650
              51        04    AE          6E   C3 00067 SUBL3   MEMBOUNDS, MEMBOUNDS+4, R1                  ; 0651
                                          51   D6 0006C INCL    R1                                          ;
                          51              04   C6 0006E DIVL2   #4, R1                                     ;
                          51              03   C2 00071 SUBL2   #3, NUMLONGS                               ;
                          60              64   D0 00074 MOVL    FMEM_BLOCK_LIST, (MEMVECTOR)                ; 0658
           04 A0          51              02   78 00077 ASHL    #2, NUMLONGS, 4(MEMVECTOR)                  ; 0659
                    04    A0              0C   C0 0007C ADDL2   #12, 4(MEMVECTOR)                           ;
                          64              50   D0 00080 MOVL    MEMVECTOR, FMEM_BLOCK_LIST                  ; 0660
                          52        08    A0   9E 00083 MOVAB   8(R0), FREEBLK                              ; 0661
                          53        08 A041   DE 00087 MOVAL   8(MEMVECTOR)[NUMLONGS], ENDPTR              ; 0662
        62                00              51   F0 0008C INSV    NUMLONGS, #0, #22, (FREEBLK)                ; 0670
                    02    A2        40    8F   8A 00091 BICB2   #64, 2(FREEBLK)                            ; 0671
                    02    A2        80    8F   88 00096 BISB2   #128, 2(FREEBLK)                           ; 0672
                    03    A2        B2    8F   90 0009B MOVB    #-78, 3(FREEBLK)                            ; 0673
     FC A3                16              00   EF 000A0 EXTZV   #0, #22, (FREEBLK), -4(ENDPTR)             ; 0674
                          50        90    A4   D0 000A6 MOVL    DBG$FREE_LIST, R0                           ; 0675
```

```
                        51      04      A0  D0 000AA        MOVL    4(R0), FORWPTR                          ;
                  04    A2              51  D0 000AE        MOVL    FORWPTR, 4(FREEBLK)                     ; 0677
                  08    A2              50  D0 000B2        MOVL    BACKPTR, 8(FREEBLK)                     ; 0678
                  08    A1              52  D0 000B6        MOVL    FREEBLK, 8(FORWPTR)                     ; 0679
                  04    A0              52  D0 000BA        MOVL    FREEBLK, 4(BACKPTR)                     ; 0680
        63      16              00      01  F0 000BE        INSV    #1, #0, #22, (ENDPTR)                   ; 0687
                  02    A3      40      8F  88 000C3        BISB2   #64, 2(ENDPTR)                          ; 0688
                  02    A3      80      8F  8A 000C8        BICB2   #128, 2(ENDPTR)                         ; 0689
                  03    A3      B2      8F  90 000CD        MOVB    #-78, 3(ENDPTR)                         ; 0690
                        50              01  D0 000D2        MOVL    #1, R0                                  ; 0691
                                            04 000D5        RET                                             ; 0693

; Routine Size:  214 bytes,    Routine Base:  DBG$CODE + 0283
```

```
;  567          0694  1 GLOBAL ROUTINE DBG$FREE_MEM_LEFT =
;  568          0695  1 !
;  569          0696  1 ! FUNCTION
;  570          0697  1 !       This routine returns the amount of free memory left in the free memory
;  571          0698  1 !       pool.  This is done by searching the entire free-list and adding up the
;  572          0699  1 !       sizes of all the free blocks.
;  573          0700  1 !
;  574          0701  1 ! INPUTS
;  575          0702  1 !       NONE
;  576          0703  1 !
;  577          0704  1 ! OUTPUTS
;  578          0705  1 !       The amount of free memory left in longwords is returned as the value.
;  579          0706  1 !
;  580          0707  1
;  581          0708  2     BEGIN
;  582          0709  2
;  583          0710  2     LOCAL
;  584          0711  2         BLKPTR: REF FMEM$BLOCK,         ! Pointer to the current free block
;  585          0712  2         MEMLEFT;                        ! The amount of free memory in the pool
;  586          0713  2
;  587          0714  2
;  588          0715  2
;  589          0716  2     ! Search the free-list and add up the sizes of all the blocks.  Then return.
;  590          0717  2     !
;  591          0718  2     MEMLEFT = 0;
;  592          0719  2     BLKPTR = .DBG$FREE_LIST[FMEM$L_FLINK];
;  593          0720  2     WHILE .BLKPTR NEQ .DBG$FREE_LIST DO
;  594          0721  3         BEGIN
;  595          0722  3         MEMLEFT = .MEMLEFT + .BLKPTR[FMEM$V_LENGTH];
;  596          0723  3         BLKPTR = .BLKPTR[FMEM$L_FLINK];
;  597          0724  2         END;
;  598          0725  2
;  599          0726  2     RETURN .MEMLEFT;
;  600          0727  2
;  601          0728  1     END;


                                                000C 00000          .ENTRY  DBG$FREE_MEM_LEFT, Save R2,R3          ; 0694
                                            52  D4 00002          CLRL    MEMLEFT                                ; 0718
                                51 00000000'    EF  D0 00004          MOVL    DBG$FREE_LIST, R1                      ; 0719
                                50        04    A1  D0 0000B          MOVL    4(R1), BLKPTR                          ;
                                51              50  D1 0000F 1$:     CMPL    BLKPTR, R1                             ; 0720
                                                0E  13 00012          BEQL    2$                                     ;
                    53          60          16      00  EF 00014          EXTZV   #0, #22, (BLKPTR), R3                  ; 0722
                                52              53  C0 00019          ADDL2   R3, MEMLEFT                            ;
                                50        04    A0  D0 0001C          MOVL    4(BLKPTR), BLKPTR                      ; 0723
                                                ED  11 00020          BRB     1$                                     ; 0720
                                50              52  D0 00022 2$:     MOVL    MEMLEFT, R0                            ; 0726
                                                    04 00025          RET                                            ; 0728

; Routine Size:  38 bytes,    Routine Base:  DBG$CODE + 0359
```

```
;  603     0729  1 GLOBAL ROUTINE DBG$GET_MEMORY(SIZE) =
;  604     0730  1 !
;  605     0731  1 ! FUNCTION
;  606     0732  1 !       This routine allocates a memory block of a specified size and returns
;  607     0733  1 !       the block's address to the caller.  The block contents is zeroed out
;  608     0734  1 !       before being returned.  This is the primary routine for getting memory
;  609     0735  1 !       blocks from the free memory pool.
;  610     0736  1 !
;  611     0737  1 !       The routine uses the First-Fit algorithm for finding a free memory
;  612     0738  1 !       block.  The free-list is thus searched until a free block which is
;  613     0739  1 !       large enough is encountered.  That block is then split into two pieces,
;  614     0740  1 !       one to be allocated to the caller and one to remain on the free list.
;  615     0741  1 !       This splitting does not occur unless the remainder is large enough to
;  616     0742  1 !       be a free block (four longwords minimum).
;  617     0743  1 !
;  618     0744  1 !       If no free block of adequate size is found, an attempt is made to remove
;  619     0745  1 !       the Run-Time Symbol Table (RST) of the Least Recently Used module.  If
;  620     0746  1 !       this succeeds (i.e., if there is such a module and it's not the only RST
;  621     0747  1 !       module), the free-list search is tried again.  This is repeated until
;  622     0748  1 !       there are no more modules to be released except the Most Recently Used
;  623     0749  1 !       one.  For this reason, the Debugger should never run out of free memory
;  624     0750  1 !       under normal circumstances--the used memory is instead recycled.
;  625     0751  1 !
;  626     0752  1 !       However, if a free block of adequate size still cannot be found, an
;  627     0753  1 !       error is signalled unless a second parameter is present.  If a second
;  628     0754  1 !       parameter is present (e.g., ADDR = DBG$GET_MEMORY(.SIZE, 0);), a zero
;  629     0755  1 !       is returned as the routine value.
;  630     0756  1 !
;  631     0757  1 ! INPUTS
;  632     0758  1 !       SIZE    - The size of the desired memory block in longwords.  This is
;  633     0759  1 !                 is the number of longwords to be allocated.
;  634     0760  1 !
;  635     0761  1 !       An optional second parameter specifies that a zero should be returned
;  636     0762  1 !                 if no block can be found to accommodate the request.  The
;  637     0763  1 !                 actual value of the parameter is not significant.  If this
;  638     0764  1 !                 parameter is omitted, an error is signalled.
;  639     0765  1 !
;  640     0766  1 ! OUTPUTS
;  641     0767  1 !       A memory block of the desired size is allocated and its address is
;  642     0768  1 !                 returned as the routine's value.  If a second parameter is
;  643     0769  1 !                 specified and no block is found, zero is returned.
;  644     0770  1 !
;  645     0771  1
;  646     0772  2     BEGIN
;  647     0773  2
;  648     0774  2     BUILTIN
;  649     0775  2         ACTUALCOUNT;                    ! Actual number of calling parameters
;  650     0776  2
;  651     0777  2     LOCAL
;  652     0778  2         BACKPTR: REF FMEM$BLOCK,        ! Pointer to previous block on free-list
;  653     0779  2         BLKPTR:  REF FMEM$BLOCK,        ! Pointer to the current memory block
;  654     0780  2         FORWPTR: REF FMEM$BLOCK,        ! Pointer to the next block on free-list
;  655     0781  2         FREEBLK: REF FMEM$BLOCK,        ! Pointer to free block split off from
;  656     0782  2                                         !      the allocated block
;  657     0783  2         LENGTH,                         ! The length of the allocated block
;  658     0784  2                                         !      including the control longword
;  659     0785  2         LRUMPTR: REF LRUM$ENTRY,        ! Pointer to the Least Recently Used
```

```
;  660          0786  2                    !       Module (LRUM) table entry
;  661          0787  2        NEXTBLK: REF FMEM$BLOCK;    ! Pointer to the next sequential block
;  662          0788  2                    !       in the memory pool
;  663          0789  2
;  664          0790  2
;  665          0791  2
;  666          0792  2        ! Make sure the requested block size is strictly positive and not too large.
;  667          0793  2        ! If less than three longwords are requested, allocate three longwords so
;  668          0794  2        ! that the block can be released to give a four longword free block.
;  669          0795  2        !
;  670          0796  2        IF .SIZE LEQ 0
;  671          0797  2        THEN
;  672          0798  2            $DBG_ERROR('GETMEMORY\DBG$GET_MEMORY');
;  673          0799  2        IF .SIZE GTR 16000
;  674          0800  2        THEN
;  675          0801  2            $DBG_ERROR('GETMEMORY\DBG$GET_MEMORY');
;  676          0802  2        LENGTH = .SIZE + 1;
;  677          0803  2        IF .LENGTH LSS 4 THEN LENGTH = 4;
;  678          0804  2
;  679          0805  2
;  680          0806  2        ! Loop through the entire free-list, searching for a block large enough to
;  681          0807  2        ! allocate to the caller.
;  682          0808  2        !
;  683          0809  2        BLKPTR = .DBG$FREE_LIST[FMEM$L_FLINK];
;  684          0810  2        WHILE TRUE DO
;  685          0811  3            BEGIN
;  686          0812  3
;  687          0813  3
;  688          0814  3            ! Do a few checks on the integrity of the free-list.
;  689          0815  3            !
;  690          0816  3            IF (.BLKPTR[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL) OR
;  691          0817  4               (.BLKPTR[FMEM$V_THISALLOC])
;  692          0818  3            THEN
;  693          0819  3                SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  694          0820  3
;  695          0821  3
;  696          0822  3            ! If the current block is large enough, allocate all or part of it.
;  697          0823  3            !
;  698          0824  3            IF .BLKPTR[FMEM$V_LENGTH] GEQ .LENGTH
;  699          0825  3            THEN
;  700          0826  4                BEGIN
;  701          0827  4
;  702          0828  4
;  703          0829  4                ! This block is large enough.  Unlink it from the free-list.
;  704          0830  4                !
;  705          0831  4                FORWPTR = .BLKPTR[FMEM$L_FLINK];
;  706          0832  4                BACKPTR = .BLKPTR[FMEM$L_BLINK];
;  707          0833  4                FORWPTR[FMEM$L_BLINK] = .BACKPTR;
;  708          0834  4                BACKPTR[FMEM$L_FLINK] = .FORWPTR;
;  709          0835  4
;  710          0836  4
;  711          0837  4                ! If there was only one block on the free list, and if
;  712          0838  4                ! that block is not large enough to be split up, then
;  713          0839  4                ! we want to return "no free storage". This is because we
;  714          0840  4                ! cannot allocate the last block on the free list (there
;  715          0841  4                ! is no way to unlink it from the free list).
;  716          0842  4                !
```

```
;  717          0843  4          IF (.FORWPTR EQL .BLKPTR) AND
;  718          0844  4             (.BACKPTR EQL .BLKPTR) AND
;  719          0845  5             (.BLKPTR[FMEM$V_LENGTH] LSS .LENGTH + 4)
;  720          0846  4          THEN
;  721          0847  4              EXITLOOP;
;  722          0848  4
;  723          0849  4
;  724          0850  4          ! If the block is large enough to accommodate the request, one
;  725          0851  4          ! control longword, and a four longword free block, split it up.
;  726          0852  4          ! Put the free portion of the block back on the free-list.
;  727          0853  4          !
;  728          0854  4          IF .BLKPTR[FMEM$V_LENGTH] GEQ .LENGTH + 4
;  729          0855  4          THEN
;  730          0856  5              BEGIN
;  731          0857  5              FREEBLK = .BLKPTR + 4*.LENGTH;
;  732          0858  5              FREEBLK[FMEM$V_LENGTH] = .BLKPTR[FMEM$V_LENGTH] - .LENGTH;
;  733          0859  5              FREEBLK[FMEM$V_THISALLOC] = FALSE;
;  734          0860  5              FREEBLK[FMEM$B_SENTINEL] = FMEM$K_SENTINEL;
;  735          0861  5              FREEBLK[FMEM$L_FLINK] = .FORWPTR;
;  736          0862  5              FREEBLK[FMEM$L_BLINK] = .BACKPTR;
;  737          0863  5              FORWPTR[FMEM$L_BLINK] = .FREEBLK;
;  738          0864  5              BACKPTR[FMEM$L_FLINK] = .FREEBLK;
;  739          0865  5              NEXTBLK = .FREEBLK + 4*.FREEBLK[FMEM$V_LENGTH];
;  740          0866  5              NEXTBLK[FMEM$L_PREVLEN] = .FREEBLK[FMEM$V_LENGTH];
;  741          0867  5              BLKPTR[FMEM$V_LENGTH] = .LENGTH;
;  742          0868  4              END;
;  743          0869  4
;  744          0870  4
;  745          0871  4              ! Mark the block as being allocated, zero it out, and return its
;  746          0872  4              ! address to the caller.
;  747          0873  4              !
;  748          0874  4              BLKPTR[FMEM$V_THISALLOC] = TRUE;
;  749          0875  4              NEXTBLK = .BLKPTR + 4*.BLKPTR[FMEM$V_LENGTH];
;  750          0876  4              NEXTBLK[FMEM$V_PREVALLOC] = TRUE;
;  751          0877  4              CH$FILL(0, 4*(.BLKPTR[FMEM$V_LENGTH] - 1), BLKPTR[FMEM$A_ALLOCBLK]);
;  752          0878  4              RETURN BLKPTR[FMEM$A_ALLOCBLK];
;  753          0879  3              END;
;  754          0880  3
;  755          0881  3
;  756          0882  3          ! The block was not large enough--link on to the next free-list entry.
;  757          0883  3          ! If this puts us past the end of the free-list exit the search loop.
;  758          0884  3          !
;  759          0885  3          BLKPTR = .BLKPTR[FMEM$L_FLINK];
;  760          0886  3          IF .BLKPTR EQL .DBG$FREE_LIST THEN EXITLOOP;
;  761          0887  2          END;
;  762          0888  2
;  763          0889  2
;  764          0890  2      ! We could not find a block large enough for the request.  If we are
;  765          0891  2      ! in a state where it is OK to expand the memory pool (either the user
;  766          0892  2      ! program has not yet run or the user has requested SET MODULE/ALLOCATE)
;  767          0893  2      ! then expand the memory pool and try again. We expand by the desired
;  768          0894  2      ! length plus 8000 longwords.
;  769          0895  2      !
;  770          0896  2      IF .DBG$GV_CONTROL[DBG$V_CONTROL_ALLOCATE] OR
;  771          0897  3         (NOT .DBG$GV_CONTROL[DBG$V_CONTROL_URUN])
;  772          0898  2      THEN
;  773          0899  3          BEGIN
```

```
;  774          0900  3          IF DBG$EXPAND_MEMORY(.LENGTH + 8000)
;  775          0901  3          THEN
;  776          0902  3              RETURN DBG$GET_MEMORY(.SIZE);
;  777          0903  2          END;
;  778          0904  2
;  779          0905  2
;  780          0906  2      ! We could not find a block large enough for the request.  If there is no
;  781          0907  2      ! second parameter, signal the error.   Otherwise return zero as the block
;  782          0908  2      ! address.
;  783          0909  2      !
;  784          0910  2      IF ACTUALCOUNT() EQL 1 THEN SIGNAL(DBG$_NOFREE);
;  785          0911  2      RETURN 0;
;  786          0912  2
;  787          0913  1      END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

24  47  42  44  5C  59  52  4F  4D  45  4D  54  45  47  18  0001C P.AAB:  .ASCII  <24>\GETMEMORY\<92>\DBG$GET_MEMORY\          ;
                    59  52  4F  4D  45  4D  5F  54  45  47  00028                                                            ;
24  47  42  44  5C  59  52  4F  4D  45  4D  54  45  47  18  00035 P.AAC:  .ASCII  <24>\GETMEMORY\<92>\DBG$GET_MEMORY\          ;
                    59  52  4F  4D  45  4D  5F  54  45  47  00044                                                            ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                    0FFC 00000          .ENTRY  DBG$GET_MEMORY, Save R2,R3,R4,R5,R6,R7,R8,-                 ; 0729
                                                                R9,R10,R11
                                04  AC  D5 00002          TSTL    SIZE                                                        ; 0796
                                    15  14 00005          BGTR    1$
                           00000000'EF  9F 00007          PUSHAB  P.AAB                                                       ; 0798
                                    01  DD 0000D          PUSHL   #1
                           00028362 8F  DD 0000F          PUSHL   #164706
                  00000000G  00         03  FB 00015     CALLS   #3, LIB$SIGNAL
                  00003E80   8F   04    AC  D1 0001C 1$:  CMPL    SIZE, #16000                                              ; 0799
                                    15  15 00024          BLEQ    2$
                           00000000'EF  9F 00026          PUSHAB  P.AAC                                                       ; 0801
                                    01  DD 0002C          PUSHL   #1
                           00028362 8F  DD 0002E          PUSHL   #164706
                  00000000G  00         03  FB 00034     CALLS   #3, LIB$SIGNAL
              56        04   AC         01  C1 0003B 2$:  ADDL3   #1, SIZE, LENGTH                                          ; 0802
                         04             56  D1 00040     CMPL    LENGTH, #4                                                ; 0803
                                        03  18 00043     BGEQ    3$
                         56             04  D0 00045     MOVL    #4, LENGTH
                         50 00000000'   EF  D0 00048 3$:  MOVL    DBG$FREE_LIST, R0                                         ; 0809
                         57        04   A0  D0 0004F     MOVL    4(R0), B[KPTR
                      B2 8F        03   A7  91 00053 4$:  CMPB    3(BLKPTR), #178                                           ; 0816
                                        04  12 00058     BNEQ    5$
                  11          67        16  E1 0005A     BBC     #22, (BLKPTR), 6$                                         ; 0817
                                        57  DD 0005E 5$:  PUSHL   BLKPTR                                                    ; 0819
                                        01  DD 00060     PUSHL   #1
                           00028AF4 8F  DD 00062          PUSHL   #166644
                  00000000G  00         03  FB 00068     CALLS   #3, LIB$SIGNAL
          56      67         16        00  ED 0006F 6$:  CMPZV   #0, #22, (BLKPTR), LENGTH                                  ; 0824
                                        03  18 00074     BGEQ    7$
```

```
                                   0089  31 00076      BRW     10$
                      59      04     A7  7D 00079 7$:  MOVQ    4(BLKPTR), FORWPTR                       ; 0831
                08    A9             5A  D0 0007D      MOVL    BACKPTR, 8(FORWPTR)                      ; 0833
                04    AA             59  D0 00081      MOVL    FORWPTR, 4(BACKPTR)                      ; 0834
                      57             59  D1 00085      CMPL    FORWPTR, BLKPTR                          ; 0843
                                     10  12 00088      BNEQ    8$
                      57             5A  D1 0008A      CMPL    BACKPTR, BLKPTR                          ; 0844
                                     0B  12 0008D      BNEQ    8$
                      50      04     A6  9E 0008F      MOVAB   4(R6), R0                                ; 0845
  50        67        16             00  ED 00093      CMPZV   #0, #22, (BLKPTR), R0
                                     78  19 00098      BLSS    11$
                      50      04     A6  9E 0009A 8$:  MOVAB   4(R6), R0                                ; 0854
  50        67        16             00  ED 0009E      CMPZV   #0, #22, (BLKPTR), R0
                                     39  19 000A3      BLSS    9$
                      58           6746  DE 000A5      MOVAL   (BLKPTR)[LENGTH], FREEBLK                ; 0857
  50        67        16             00  EF 000A9      EXTZV   #0, #22, (BLKPTR), R0                    ; 0858
                      50             56  C2 000AE      SUBL2   LENGTH, R0
  68        16        00             50  F0 000B1      INSV    R0, #0, #22, (FREEBLK)
                02    A8      40     8F  8A 000B6      BICB2   #64, 2(FREEBLK)                          ; 0859
                03    A8      B2     8F  90 000BB      MOVB    #-78, 3(FREEBLK)                         ; 0860
                04    A8             59  7D 000C0      MOVQ    FORWPTR, 4(FREEBLK)                      ; 0861
                08    A9             58  D0 000C4      MOVL    FREEBLK, 8(FORWPTR)                      ; 0863
                04    AA             58  D0 000C8      MOVL    FREEBLK, 4(BACKPTR)                      ; 0864
  50        68        16             00  EF 000CC      EXTZV   #0, #22, (FREEBLK), R0                   ; 0865
                      5B           6840  DE 000D1      MOVAL   (FREEBLK)[R0], NEXTBLK
                FC    AB             50  D0 000D5      MOVL    R0, -4(NEXTBLK)                          ; 0866
  67        16        00             56  F0 000D9      INSV    LENGTH, #0, #22, (BLKPTR)                ; 0867
                02    A7      40     8F  88 000DE 9$:  BISB2   #64, 2(BLKPTR)                           ; 0874
  50        67        16             00  EF 000E3      EXTZV   #0, #22, (BLKPTR), R0                    ; 0875
                      5B           6740  DE 000E8      MOVAL   (BLKPTR)[R0], NEXTBLK
                02    AB      80     8F  88 000EC      BISB2   #128, 2(NEXTBLK)                         ; 0876
                                     50  D7 000F1      DECL    R0                                       ; 0877
                      50             04  C4 000F3      MULL2   #4, R0
  50        00        6E             00  2C 000F6      MOVC5   #0, (SP), #0, R0, 4(BLKPTR)
                              04     A7     000FB
                      50      04     A7  9E 000FD      MOVAB   4(BLKPTR), R0                            ; 0878
                                         04 00101      RET
                      57      04     A7  D0 00102 10$: MOVL    4(BLKPTR), BLKPTR                        ; 0885
            00000000' EF             57  D1 00106      CMPL    BLKPTR, DBG$FREE_LIST                    ; 0886
                                     03  13 0010D      BEQL    11$
                                   FF41  31 0010F      BRW     4$
                           00000000G 00  95 00112 11$: TSTB    DBG$GV_CONTROL                           ; 0896
                                     08  19 00118      BLSS    12$
         15 00000000G 00             03  E0 0011A      BBS     #3, DBG$GV_CONTROL, 13$                  ; 0897
                            1F40     C6  9F 00122 12$: PUSHAB  8000(LENGTH)                             ; 0900
                 FDD9 CF             01  FB 00126      CALLS   #1, DBG$EXPAND_MEMORY
                      09             50  E9 0012B      BLBC    R0, 13$
                              04     AC  DD 0012E      PUSHL   SIZE                                     ; 0902
                 FECA CF             01  FB 00131      CALLS   #1, DBG$GET_MEMORY
                                         04 00136      RET
                      01             6C  91 00137 13$: CMPB    (AP), #1                                 ; 0910
                                     0D  12 0013A      BNEQ    14$
                         00028332    8F  DD 0013C      PUSHL   #164658
            00000000G 00             01  FB 00142      CALLS   #1, LIB$SIGNAL
                                     50  D4 00149 14$: CLRL    R0                                       ; 0911
                                         04 0014B      RET                                              ; 0913
```

; Routine Size: 332 bytes,   Routine Base: DBG$CODE + 037F

```
;  789          0914  1 GLOBAL ROUTINE DBG$GET_TEMPMEM(LENGTH) =
;  790          0915  1 !
;  791          0916  1 ! FUNCTIONAL DESCRIPTION:
;  792          0917  1 !       This routine allocates a "temporary" memory block from the memory pool
;  793          0918  1 !       and returns its address.  A "temporary" block is a memory block which
;  794          0919  1 !       automatically disappears the next time DBG$REL_TEMPMEM is called, norm-
;  795          0920  1 !       ally at the end of the current command.  The advantage of temporary
;  796          0921  1 !       blocks is that they do not have to be explicitly released to the memory
;  797          0922  1 !       pool--one call on DBG$REL_TEMPMEM releases all temporary blocks.
;  798          0923  1 !
;  799          0924  1 !       All temporary memory blocks are put on the singly linked list pointed
;  800          0925  1 !       to by DBG$TEMP_MEMORY.  To accommodate this link and a second control
;  801          0926  1 !       word with the block's length and a sentinel value, two extra longwords
;  802          0927  1 !       are needed.
;  803          0928  1 !
;  804          0929  1 ! INPUTS
;  805          0930  1 !       LENGTH  - The desired length of the temporary block in longwords.
;  806          0931  1 !
;  807          0932  1 ! OUTPUTS
;  808          0933  1 !       The requested block is allocated and put on the temporary memory list.
;  809          0934  1 !               The address of the allocated block is returned as the rout-
;  810          0935  1 !               ine's value.
;  811          0936  1 !
;  812          0937  1
;  813          0938  2     BEGIN
;  814          0939  2
;  815          0940  2     LOCAL
;  816          0941  2         BLKADDR: REF VECTOR[,LONG],     ! Pointer to the block allocated by
;  817          0942  2                                         !       routine DBG$GET_MEMORY
;  818          0943  2         BLKPTR: REF FMEM$BLOCK;         ! Pointer to the temporary block's own
;  819          0944  2                                         !       control longword
;  820          0945  2
;  821          0946  2
;  822          0947  2
;  823          0948  2     ! Allocate a memory block of the desired size plus two longwords.  Link this
;  824          0949  2     ! block into the temporary block list using longword 0, fill some control
;  825          0950  2     ! information into longword 1, and return the address of longword 2.
;  826          0951  2     !
;  827          0952  2     BLKADDR = DBG$GET_MEMORY(.LENGTH + 2);
;  828          0953  2     BLKADDR[0] = .DBG$TEMP_MEMORY;
;  829          0954  2     DBG$TEMP_MEMORY = .BLKADDR;
;  830          0955  2     BLKPTR = BLKADDR[1];
;  831          0956  2     BLKPTR[FMEM$V_LENGTH] = .LENGTH + 1;
;  832          0957  2     BLKPTR[FMEM$V_THISALLOC] = TRUE;
;  833          0958  2     BLKPTR[FMEM$V_PREVALLOC] = TRUE;
;  834          0959  2     BLKPTR[FMEM$B_SENTINEL] = FMEM$K_TEMPSENT;
;  835          0960  2     RETURN BLKPTR[FMEM$A_ALLOCBLK];
;  836          0961  2
;  837          0962  1     END;

                                0004 00000          .ENTRY  DBG$GET_TEMPMEM, Save R2                        ; 0914
                    52 00000000'  EF   9E 00002     MOVAB   DBG$TEMP_MEMORY, R2                             :
          7E        04    AC        02   C1 00009     ADDL3   #2, LENGTH, -(SP)                               ; 0952
```

```
                        FEA1    CF              01  FB 0000E    CALLS   #1, DBG$GET_MEMORY                      ;
                                60              62  D0 00013    MOVL    DBG$TEMP_MEMORY, (BLKADDR)              ; 0953
                                62              80  DE 00016    MOVAL   (BLKADDR)+, DBG$TEMP_MEMORY             ; 0954
                    51          04      AC      01  C1 00019    ADDL3   #1, LENGTH, R1                          ; 0956
            80      16                  00      51  F0 0001E    INSV    R1, #0, #22, (BLKPTR)+                  ;
                                01      A0  C0  8F  88 00023    BISB2   #192, 1(BLKPTR)                         ; 0958
                                02      A0  B4  8F  90 00028    MOVB    #-76, 2(BLKPTR)                         ; 0959
                                        50      03  C0 0002D    ADDL2   #3, R0                                  ; 0960
                                                    04 00030    RET                                             ; 0962

; Routine Size:  49 bytes,    Routine Base:  DBG$CODE + 04CB
```

```
;  839          0963  1 GLOBAL ROUTINE DBG$INIT_MEMORY: NOVALUE =
;  840          0964  1 !
;  841          0965  1 ! FUNCTION
;  842          0966  1 !       This routine initializes the free memory pool from which DBG$GET_MEMORY
;  843          0967  1 !       allocates memory blocks.  This is done by getting space either up in
;  844          0968  1 !       high P1 space (for a normal or a testable Debugger) or from $EXPREG
;  845          0969  1 !       (for a Super-Debugger).  The free memory pool is initialized to have a
;  846          0970  1 !       free-list list head and one big free memory block from which space can
;  847          0971  1 !       later be allocated.
;  848          0972  1 !
;  849          0973  1 !       Note that the memory pool can be expanded later by calls on the routine
;  850          0974  1 !       DBG$EXPAND_MEMORY.
;  851          0975  1 !
;  852          0976  1 ! INPUTS
;  853          0977  1 !       NONE
;  854          0978  1 !
;  855          0979  1 ! OUTPUTS
;  856          0980  1 !       NONE
;  857          0981  1 !
;  858          0982  1
;  859          0983  2     BEGIN
;  860          0984  2
;  861          0985  2     LOCAL
;  862          0986  2         ENDPTR: REF FMEM$BLOCK,          ! Pointer to the end of memory pool area
;  863          0987  2         FREEBLK: REF FMEM$BLOCK,         ! Pointer to one big free block in pool
;  864          0988  2         LENGTH,                          ! Longword length of memory pool area
;  865          0989  2         LISTHEAD: REF FMEM$BLOCK,        ! Pointer to the free-list list head
;  866          0990  2         NUMBYTES,                        ! Number of bytes to get from $EXPREG
;  867          0991  2         MEMBOUNDS: VECTOR[2, LONG],      ! Start and end addresses of new area
;  868          0992  2         MEMVECTOR: REF VECTOR[,LONG],    ! Pointer to acquired memory pool area
;  869          0993  2         NUMPAGES,                        ! Number of pages to get from $EXPREG
;  870          0994  2         STATUS: BLOCK[1, LONG];          ! Status code from $CRETVA or $EXPREG
;  871          0995  2
;  872          0996  2
;  873          0997  2
;  874          0998  2     ! If this is a normal Debugger or a Testable Debugger, we allocate the
;  875          0999  2     ! initial chunk of memory pool area up in high user memory (P1 space).
;  876          1000  2     ! This is done through the Create Virtual Address Space system service.
;  877          1001  2     !
;  878          1002  2     IF NOT .DBG$GV_CONTROL[DBG$V_CONTROL_SDBG]
;  879          1003  2     THEN
;  880          1004  3         BEGIN
;  881          1005  3         MEMBOUNDS[0] = %X'7FFF0000';
;  882          1006  3         MEMBOUNDS[1] = %X'7FFFFFFF';
;  883          1007  3         STATUS = $CRETVA(INADR=MEMBOUNDS, RETADR=MEMBOUNDS);
;  884          1008  3         IF NOT .STATUS
;  885          1009  3         THEN
;  886          1010  4             BEGIN
;  887          1011  4             STATUS[STS$V_SEVERITY] = STS$K_SEVERE;
;  888          1012  4             SIGNAL(.STATUS);
;  889          1013  3             END;
;  890          1014  3
;  891          1015  3         END
;  892          1016  3
;  893          1017  3
;  894          1018  3     ! If this is a Super-Debugger, we get the initial chunk of memory from the
;  895          1019  3     ! $EXPREG system service.  We cannot touch the P1-space area because it is
```

```
;  896          1020  3      ! used by the Testable Debugger that the Super-Debugger is debugging.
;  897          1021  3      !
;  898          1022  2      ELSE
;  899          1023  3          BEGIN
;  900          1024  3          NUMBYTES = 65536;
;  901          1025  3          NUMPAGES = .NUMBYTES/512;
;  902          1026  3          STATUS = $EXPREG(PAGCNT=.NUMPAGES, RETADR=MEMBOUNDS);
;  903          1027  3          IF NOT .STATUS THEN SIGNAL(.STATUS);
;  904          1028  2          END;
;  905          1029  2
;  906          1030  2      ! Now initialize the memory pool area.  The first longword is a forward link
;  907          1031  2      ! to the next memory pool area--for this initial area this field is always
;  908          1032  2      ! zero.  The second longword contains the byte length of this area.  Long-
;  909          1033  2      ! words 2 - 5 then contain the free-list list head.  The rest of the area
;  910          1034  2      ! from longword 6 through the next to last longword constitutes a big free
;  911          1035  2      ! block available for allocation.  Finally, the last longword is the termi-
;  912          1036  2      ! nator block for this memory pool area.
;  913          1037  2      !
;  914          1038  2      MEMVECTOR = .MEMBOUNDS[0];
;  915          1039  2      LENGTH = (.MEMBOUNDS[1] - .MEMBOUNDS[0] + 1)/%UPVAL;
;  916          1040  2      MEMVECTOR[0] = 0;
;  917          1041  2      MEMVECTOR[1] = .LENGTH*%UPVAL;
;  918          1042  2      LISTHEAD = MEMVECTOR[2];
;  919          1043  2      FREEBLK = MEMVECTOR[6];
;  920          1044  2      ENDPTR = MEMVECTOR[.LENGTH - 1];
;  921          1045  2
;  922          1046  2
;  923          1047  2      ! Set up the free-list list head.
;  924          1048  2      !
;  925          1049  2      LISTHEAD[FMEM$V_LENGTH] = 4;
;  926          1050  2      LISTHEAD[FMEM$V_THISALLOC] = FALSE;
;  927          1051  2      LISTHEAD[FMEM$V_PREVALLOC] = TRUE;
;  928          1052  2      LISTHEAD[FMEM$B_SENTINEL] = FMEM$K_SENTINEL;
;  929          1053  2      LISTHEAD[FMEM$L_FLINK] = .FREEBLK;
;  930          1054  2      LISTHEAD[FMEM$L_BLINK] = .FREEBLK;
;  931          1055  2      FREEBLK[FMEM$L_PREVLEN] = 4;
;  932          1056  2
;  933          1057  2
;  934          1058  2      ! Set up the one big free block in the initial memory pool.  Note that we
;  935          1059  2      ! claim that the List Head is allocated--this prevents the List Head from
;  936          1060  2      ! being coalesced with the following memory block by DBG$REL_MEMORY.
;  937          1061  2      !
;  938          1062  2      FREEBLK[FMEM$V_LENGTH] = .LENGTH - 7;
;  939          1063  2      FREEBLK[FMEM$V_THISALLOC] = FALSE;
;  940          1064  2      FREEBLK[FMEM$V_PREVALLOC] = TRUE;
;  941          1065  2      FREEBLK[FMEM$B_SENTINEL] = FMEM$K_SENTINEL;
;  942          1066  2      FREEBLK[FMEM$L_FLINK] = .LISTHEAD;
;  943          1067  2      FREEBLK[FMEM$L_BLINK] = .LISTHEAD;
;  944          1068  2      ENDPTR[FMEM$L_PREVLEN] = .FREEBLK[FMEM$V_LENGTH];
;  945          1069  2
;  946          1070  2
;  947          1071  2      ! Set up the "allocated" terminator block at the end of the memory pool
;  948          1072  2      ! area to prevent blocks from being coalesced beyond the end of the area.
;  949          1073  2      !
;  950          1074  2      ENDPTR[FMEM$V_LENGTH] = 1;
;  951          1075  2      ENDPTR[FMEM$V_THISALLOC] = TRUE;
;  952          1076  2      ENDPTR[FMEM$V_PREVALLOC] = FALSE;
```

```
2       ENDPTR[FMEM$B_SENTINEL] = FMEM$K_SENTINEL;
2
2       ! Set up the two OWN pointers we need to maintain.  One points the the list
2       ! of memory pool areas and the other points to the free-list.  Then return.
2       !
2       FMEM_BLOCK_LIST = MEMVECTOR[0];
2       DBG$FREE_LIST = .LISTHEAD;
2       RETURN;
2
1       END;


                                                                   .EXTRN  SYS$CRETVA

                                           003C 00000              .ENTRY  DBG$INIT_MEMORY, Save R2,R3,R4,R5           : 0963
                   5E                08      C2 00002              SUBL2   #8, SP
     28 00000000G  00                01      E0 00005              BBS     #1, DBG$GV_CONTROL, 1$                      : 1002
                   6E 7FFF0000       8F      D0 0000D              MOVL    #2147418112, MEMBOUNDS                      : 1005
             04    AE 7FFFFFFF       8F      D0 00014              MOVL    #2147483647, MEMBOUNDS+4                    : 1006
                                     7E      D4 0001C              CLRL    -(SP)                                       : 1007
                              04     AE      9F 0001E              PUSHAB  MEMBOUNDS
                              08     AE      9F 00021              PUSHAB  MEMBOUNDS
        00000000G  00                03      FB 00024              CALLS   #3, SYS$CRETVA
                   30                50      E8 0002B              BLBS    STATUS, 3$                                  : 1008
 50          03    00                04      F0 0002E              INSV    #4, #0, #3, STATUS                          : 1011
                                     20      11 00033              BRB     2$                                          : 1012
                   50 00010000       8F      D0 00035 1$:          MOVL    #65536, NUMBYTES                            : 1024
        51         50 00000200       8F      C7 0003C              DIVL3   #512, NUMBYTES, NUMPAGES                    : 1025
                                     7E      7C 00044              CLRQ    -(SP)                                       : 1026
                              08     AE      9F 00046              PUSHAB  MEMBOUNDS
                                     51      DD 00049              PUSHL   NUMPAGES
        00000000G  00                04      FB 0004B              CALLS   #4, SYS$EXPREG
                   09                50      E8 00052              BLBS    STATUS, 3$                                  : 1027
                                     50      DD 00055 2$:          PUSHL   STATUS
        00000000G  00                01      FB 00057              CALLS   #1, LIB$SIGNAL
                   51                6E      D0 0005E 3$:          MOVL    MEMBOUNDS, MEMVECTOR                        : 1038
        50   04    AE                6E      C3 00061              SUBL3   MEMBOUNDS, MEMBOUNDS+4, R0                  : 1039
                                     50      D6 00066              INCL    R0
        53         50                04      C7 00068              DIVL3   #4, R0, LENGTH
                                     61      D4 0006C              CLRL    (MEMVECTOR)                                 : 1040
     04 A1         53                02      78 0006E              ASHL    #2, LENGTH, 4(MEMVECTOR)                    : 1041
                   52         08     A1      9E 00073              MOVAB   8(R1), LISTHEAD                             : 1042
                   50         18     A1      9E 00077              MOVAB   24(R1), FREEBLK                             : 1043
                   54      FC A143   DE 0007B                      MOVAL   -4(MEMVECTOR)[LENGTH], ENDPTR               : 1044
 62          16    00                04      F0 00080              INSV    #4, #0, #22, (LISTHEAD)                     : 1049
             02    A2         40     8F      8A 00085              BICB2   #64, 2(LISTHEAD)                            : 1050
             02    A2         80     8F      88 0008A              BISB2   #128, 2(LISTHEAD)                           : 1051
             03    A2         B2     8F      90 0008F              MOVB    #-78, 3(LISTHEAD)                           : 1052
             04    A2                50      D0 00094              MOVL    FREEBLK, 4(LISTHEAD)                        : 1053
             08    A2                50      D0 00098              MOVL    FREEBLK, 8(LISTHEAD)                        : 1054
             FC    A0                04      D0 0009C              MOVL    #4, -4(FREEBLK)                             : 1055
                   55         F9     A3      9E 000A0              MOVAB   -7(R3), R5                                  : 1062
 60          16    00                55      F0 000A4              INSV    R5, #0, #22, (FREEBLK)
             02    A0         40     8F      8A 000A9              BICB2   #64, 2(FREEBLK)                             : 1063
             02    A0         80     8F      88 000AE              BISB2   #128, 2(FREEBLK)                            : 1064
```

```
                        03   A0        B2   8F   90 000B3          MOVB    #-78, 3(FREEBLK)                          ; 1065
                        04   A0             52   D0 000B8          MOVL    LISTHEAD, 4(FREEBLK)                      ; 1066
                        08   A0             52   D0 000BC          MOVL    LISTHEAD, 8(FREEBLK)                      ; 1067
     FC    A4              60              16   00   EF 000C0     EXTZV   #0, #22, (FREEBLK), -4(ENDPTR)            ; 1068
           64              16              00   01   F0 000C6     INSV    #1, #0, #22, (ENDPTR)                     ; 1074
                        02   A4        40   8F   88 000CB          BISB2   #64, 2(ENDPTR)                            ; 1075
                        02   A4        80   8F   8A 000D0          BICB2   #128, 2(ENDPTR)                           ; 1076
                        03   A4        B2   8F   90 000D5          MOVB    #-78, 3(ENDPTR)                           ; 1077
                00000000'   EF             51   D0 000DA          MOVL    MEMVECTOR, FMEM_BLOCK_LIST                ; 1083
                00000000'   EF             52   D0 000E1          MOVL    LISTHEAD, DBG$FREE_LIST                   ; 1084
                                                04 000E8          RET                                               ; 1087

; Routine Size:  233 bytes,    Routine Base:  DBG$CODE + 04FC
```

```
;  965          1088  1 GLOBAL ROUTINE DBG$NPARSE_ALLOCATE (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
;  966          1089  1 !
;  967          1090  1 !  FUNCTION
;  968          1091  1 !        Parses the ALLOCATE command. The "ALLOCATE" has already been
;  969          1092  1 !        recognized by the top-level parse routine. This routine picks
;  970          1093  1 !        up the integer argument and constructs a command execution tree.
;  971          1094  1 !        The tree has the verb node for ALLOCATE, and a noun node with
;  972          1095  1 !        the integer argument.
;  973          1096  1 !
;  974          1097  1 !  INPUTS
;  975          1098  1 !        INPUT_DESC      - A string descriptor for the remaining command line
;  976          1099  1 !        VERB_NODE       - The already existing verb node.
;  977          1100  1 !        MESSAGE_VECT    - The address of a message argument vector.
;  978          1101  1 !
;  979          1102  1 !  OUTPUTS
;  980          1103  1 !        The return value is one of:
;  981          1104  1 !        STS$K_SUCCESS   - Success. A command execution tree was constructed.
;  982          1105  1 !        STS$K_SEVERE    - Failure. An error message vector is constructed.
;  983          1106  1 !
;  984          1107  2     BEGIN
;  985          1108  2
;  986          1109  2     MAP
;  987          1110  2         INPUT_DESC : REF DBG$STG_DESC,
;  988          1111  2         VERB_NODE  : REF DBG$VERB_NODE;
;  989          1112  2
;  990          1113  2     BIND
;  991          1114  2         DBG$CS_CR       = UPLIT BYTE (1, DBG$K_CAR_RETURN);
;  992          1115  2
;  993          1116  2     LOCAL
;  994          1117  2         NOUN_NODE: REF DBG$NOUN_NODE;   ! Pointer to a noun node
;  995          1118  2
;  996          1119  2     ! Check for end-of-line. This is an error.
;  997          1120  2     !
;  998          1121  2     IF .INPUT_DESC [DSC$W_LENGTH] EQL 0
;  999          1122  2     THEN
; 1000          1123  3         BEGIN
; 1001          1124  3         .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE);
; 1002          1125  3         RETURN STS$K_SEVERE;
; 1003          1126  2         END;
; 1004          1127  2     IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
; 1005          1128  2     THEN
; 1006          1129  3         BEGIN
; 1007          1130  3         .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE);
; 1008          1131  3         RETURN STS$K_SEVERE;
; 1009          1132  2         END;
; 1010          1133  2
; 1011          1134  2     ! Create and link a noun node.
; 1012          1135  2     !
; 1013          1136  2     NOUN_NODE = DBG$GET_TEMPMEM(DBG$K_NOUN_NODE_SIZE);
; 1014          1137  2     VERB_NODE [DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
; 1015          1138  2
; 1016          1139  2     ! Pick up the integer argument.
; 1017          1140  2     !
; 1018          1141  2     IF NOT DBG$NSAVE_DECIMAL_INTEGER (.INPUT_DESC,
; 1019          1142  2         NOUN_NODE [DBG$L_NOUN_VALUE],
; 1020          1143  2         .MESSAGE_VECT)
; 1021          1144  2     THEN
```

```
;  1022          1145  2          RETURN STS$K_SEVERE;
;  1023          1146  2
;  1024          1147  2      ! Return success.
;  1025          1148  2      !
;  1026          1149  2      RETURN STS$K_SUCCESS;
;  1027          1150  2
;  1028          1151  1      END;


                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

                                0D  01  0004E P.AAD:    .BYTE   1, 13                                   ;
                                                DBG$CS_CR=          P.AAD

                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                  0004 00000            .ENTRY  DBG$NPARSE_ALLOCATE, Save R2            ; 1088
                        52      04  AC  D0 00002        MOVL    INPUT_DESC, R2                          ; 1121
                                    62  B5 00006        TSTW    (R2)                                    ;
                                    14  13 00008        BEQL    1$                                      ;
                                    01  DD 0000A        PUSHL   #1                                      ; 1127
                    00000000'       EF  9F 0000C        PUSHAB  DBG$CS_CR                               ;
                                    52  DD 00012        PUSHL   R2                                      ;
              00000000G  00         03  FB 00014        CALLS   #3, DBG$NMATCH                          ;
                        13          50  E9 0001B        BLBC    R0, 2$                                  ;
                    000280D0        8F  DD 0001E 1$:    PUSHL   #164048                                 ; 1130
              00000000G  00         01  FB 00024        CALLS   #1, DBG$NMAKE_ARG_VECT                  ;
                    0C  BC          50  D0 0002B        MOVL    R0, @MESSAGE_VECT                       ;
                                    20  11 0002F        BRB     3$                                      ; 1131
                                    04  DD 00031 2$:    PUSHL   #4                                      ; 1136
                  FEAE  CF          01  FB 00033        CALLS   #1, DBG$GET_TEMPMEM                     ;
                        51      08  AC  D0 00038        MOVL    VERB_NODE, R1                           ; 1137
                    08  A1          50  D0 0003C        MOVL    NOUN_NODE, 8(R1)                        ;
                                0C  AC  DD 00040        PUSHL   MESSAGE_VECT                            ; 1143
                                    50  DD 00043        PUSHL   NOUN_NODE                               ; 1142
                                    52  DD 00045        PUSHL   R2                                      ;
              00000000G  00         03  FB 00047        CALLS   #3, DBG$NSAVE_DECIMAL_INTEGER           ;
                        04          50  E8 0004E        BLBS    R0, 4$                                  ;
                        50          04  D0 00051 3$:    MOVL    #4, R0                                  ; 1145
                                        04 00054        RET                                             ;
                        50          01  D0 00055 4$:    MOVL    #1, R0                                  ; 1149
                                        04 00058        RET                                             ; 1151

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 05E5
```

```
;  1030          1152  1 GLOBAL ROUTINE DBG$NEXECUTE_ALLOCATE (VERB_NODE, MESSAGE_VECT) =
;  1031          1153  1 !
;  1032          1154  1 ! FUNCTION
;  1033          1155  1 !       This routine executes the ALLOCATE command.
;  1034          1156  1 !
;  1035          1157  1 ! INPUTS
;  1036          1158  1 !       VERB_NODE       - The command verb that is the start of the command
;  1037          1159  1 !                         execution tree.
;  1038          1160  1 !       MESSAGE_VECT    - The address of the error message vector.
;  1039          1161  1 !
;  1040          1162  1 ! OUTPUTS
;  1041          1163  1 !       The return code is one of:
;  1042          1164  1 !       STS$K_SUCCESS   - Success. Memory was expanded.
;  1043          1165  1 !       STS$K_SEVERE    _ Failure. A message argument vector is constructed
;  1044          1166  1 !
;  1045          1167  2     BEGIN
;  1046          1168  2
;  1047          1169  2     MAP
;  1048          1170  2         VERB_NODE       : REF DBG$VERB_NODE;
;  1049          1171  2
;  1050          1172  2     LOCAL
;  1051          1173  2         NOUN_NODE: REF DBG$NOUN_NODE,   ! A pointer to the noun node.
;  1052          1174  2         NUM_BYTES;                      ! The number of bytes to expand memory
;  1053          1175  2
;  1054          1176  2     ! Obtain the noun node.
;  1055          1177  2     !
;  1056          1178  2     NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
;  1057          1179  2
;  1058          1180  2     ! Check for zero - this is an error.
;  1059          1181  2     !
;  1060          1182  2     IF .NOUN_NODE EQL 0
;  1061          1183  2     THEN
;  1062          1184  2         $DBG_ERROR ('GETMEMORY\DBG$NEXECUTE_ALLOCATE');
;  1063          1185  2
;  1064          1186  2     ! Extract the argument.
;  1065          1187  2     !
;  1066          1188  2     NUM_BYTES = .NOUN_NODE [DBG$L_NOUN_VALUE];
;  1067          1189  2
;  1068          1190  2     ! Force the user to allocate at least 1000 bytes.
;  1069          1191  2     !
;  1070          1192  2     IF .NUM_BYTES LSS 1000
;  1071          1193  2     THEN
;  1072          1194  3         BEGIN
;  1073          1195  3         .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_ALLOBNDS);
;  1074          1196  3         RETURN STS$K_SEVERE;
;  1075          1197  2         END;
;  1076          1198  2
;  1077          1199  2     ! Call DBG$EXPAND_MEMORY to get the memory.
;  1078          1200  2     !
;  1079          1201  2     IF NOT DBG$EXPAND_MEMORY ((3+.NUM_BYTES)/4)
;  1080          1202  2     THEN
;  1081          1203  3         BEGIN
;  1082          1204  3         .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_UNAEXPMEM, 1, .NUM_BYTES);
;  1083          1205  3         RETURN STS$K_SEVERE;
;  1084          1206  2         END;
;  1085          1207  2
;  1086          1208  2     ! Return success.
```

```
;  1087          1209  2     !
;  1088          1210  2     RETURN STS$K_SUCCESS;
;  1089          1211  2
;  1090          1212  1     END;


                                                                .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

24  47  42  44  5C  59  52  4F  4D  45  4D  54  45  47  1F  00050 P.AAE:  .ASCII  <31>\GETMEMORY\<92>\DBG$NEXECUTE_ALLOC\      ;
    43  4F  4C  4C  41  5F  45  54  55  43  45  58  45  4E  0005F                                                              ;
                                    45  54  41  0006D           .ASCII  \ATE\                                                 ;


                                                                .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                                  000C 00000    .ENTRY  DBG$NEXECUTE_ALLOCATE, Save R2,R3                     ; 1152
                                53 00000000G    00   9E 00002    MOVAB   DBG$NMAKE_ARG_VECT, R3                              ;
                                50         04   AC   D0 00009    MOVL    VERB_NODE, R0                                       ; 1178
                                52         08   A0   D0 0000D    MOVL    8(R0), NOUN_NODE                                    ;
                                                  15   12 00011    BNEQ    1$                                                  ; 1182
                                   00000000'    EF   9F 00013    PUSHAB  P.AAE                                               ; 1184
                                                  01   DD 00019    PUSHL   #1                                                  ;
                                   00028362     8F   DD 0001B    PUSHL   #164706                                             ;
                        00000000G   00          03   FB 00021    CALLS   #3, LIB$SIGNAL                                      ;
                                    52          62   D0 00028 1$: MOVL    (NOUN_NODE), NUM_BYTES                              ; 1188
                        000003E8    8F          52   D1 0002B    CMPL    NUM_BYTES, #1000                                    ; 1192
                                                  0B   18 00032    BGEQ    2$                                                  ;
                                   00028E88     8F   DD 00034    PUSHL   #167560                                             ; 1195
                                63              01   FB 0003A    CALLS   #1, DBG$NMAKE_ARG_VECT                              ;
                                                  1D   11 0003D    BRB     3$                                                  ;
                                50         03   A2   9E 0003F 2$: MOVAB   3(R2), R0                                        ; 1201
                             7E             50   04   C7 00043    DIVL3   #4, R0, -(SP)                                       ;
                                FBF9        CF   01   FB 00047    CALLS   #1, DBG$EXPAND_MEMORY                               ;
                                15          50   E8 0004C    BLBS    R0, 4$                                              ;
                                            52   DD 0004F    PUSHL   NUM_BYTES                                           ; 1204
                                            01   DD 00051    PUSHL   #1                                                  ;
                                   00028E80     8F   DD 00053    PUSHL   #167552                                             ;
                                63              03   FB 00059    CALLS   #3, DBG$NMAKE_ARG_VECT                              ;
                                   08       BC   50   D0 0005C 3$: MOVL    R0, @MESSAGE_VECT                                   ;
                                50              04   D0 00060    MOVL    #4, R0                                              ; 1205
                                                  04 00063    RET                                                             ;
                                50              01   D0 00064 4$: MOVL    #1, R0                                              ; 1210
                                                  04 00067    RET                                                             ; 1212

; Routine Size:  104 bytes,    Routine Base:  DBG$CODE + 063E
```

```
;  1092          1213  1 GLOBAL ROUTINE DBG$PARSE_ALLOCATE (PARSE_STG_DESC) =
;  1093          1214  1 !
;  1094          1215  1 ! FUNCTION
;  1095          1216  1 !       This routine provides an interface from the old debugger to the
;  1096          1217  1 !       new debugger parse network for ALLOCATE.
;  1097          1218  1 !
;  1098          1219  1 ! INPUTS
;  1099          1220  1 !       PARSE_STG_DESC  - A string descriptor for the remaining input
;  1100          1221  1 !
;  1101          1222  1 ! OUTPUTS
;  1102          1223  1 !       A command execution network is constructed, and a pointer to
;  1103          1224  1 !       the verb node is returned.
;  1104          1225  1 !
;  1105          1226  2     BEGIN
;  1106          1227  2
;  1107          1228  2     MAP
;  1108          1229  2         PARSE_STG_DESC: REF DBG$STG_DESC;
;  1109          1230  2
;  1110          1231  2     LOCAL
;  1111          1232  2         CHAR: BYTE,                             ! Holds a character
;  1112          1233  2         DUMMY_MESS_VECT: REF VECTOR,    ! Address of message vector returned
;  1113          1234  2                                         !     from DBG$NPARSE_ALLOCATE
;  1114          1235  2         LEN,                            ! Length of command line
;  1115          1236  2         PARSE_STG_PTR,                  ! Pointer into command line
;  1116          1237  2         STG: REF VECTOR [,BYTE],        ! Pointer into a new copy of the
;  1117          1238  2                                         !     command line
;  1118          1239  2         VERB_NODE: REF DBG$VERB_NODE;   ! Pointer to a verb node.
;  1119          1240  2
;  1120          1241  2     ! Allocate space for a verb node.
;  1121          1242  2     !
;  1122          1243  2     VERB_NODE = DBG$GET_TEMPMEM(DBG$K_VERB_NODE_SIZE);
;  1123          1244  2
;  1124          1245  2
;  1125          1246  2     ! Stuff a carriage return at the end
;  1126          1247  2     ! of the input line since this is what the new style
;  1127          1248  2     ! parser expects to see. Also, translate the line to
;  1128          1249  2     ! upper case (the new debugger does this; the old does not)
;  1129          1250  2     !
;  1130          1251  2     len = .parse_stg_desc[dsc$w_length];
;  1131          1252  2     stg = dbg$get_tempmem (1+(1+.len)/%UPVAL);
;  1132          1253  2     parse_stg_ptr = ch$ptr(.parse_stg_desc[dsc$a_pointer]);
;  1133          1254  2     INCR I FROM 0 TO .len-1 DO
;  1134          1255  3         BEGIN
;  1135          1256  3         char = ch$rchar_a(parse_stg_ptr);
;  1136          1257  3         IF .char GEQ %C'a' AND .char LEQ %C'z'
;  1137          1258  3         THEN
;  1138          1259  4             stg[.i] = .char - (%C'a'-%C'A')
;  1139          1260  3         ELSE
;  1140          1261  3             stg[.i] = .char;
;  1141          1262  2         END;
;  1142          1263  2     stg[.len] = dbg$k_car_return;
;  1143          1264  2     parse_stg_desc[dsc$a_pointer] = .stg;
;  1144          1265  2     parse_stg_desc[dsc$w_length] =
;  1145          1266  2         .parse_stg_desc[dsc$w_length] + 1;
;  1146          1267  2
;  1147          1268  2     ! Now call the parser on the remainder of the input line
;  1148          1269  2     !
```

```
;  1149          1270  2     IF NOT DBG$NPARSE_ALLOCATE (.PARSE_STG_DESC,
;  1150          1271  2         .VERB_NODE, DUMMY_MESS_VECT)
;  1151          1272  2     THEN
;  1152          1273  2
;  1153          1274  2         ! If the above routine does not return success, then we signal
;  1154          1275  2         ! an error using the error message vector that we got back.
;  1155          1276  2         !
;  1156          1277  3         BEGIN
;  1157          1278  3         EXTERNAL ROUTINE
;  1158          1279  3             LIB$SIGNAL : ADDRESSING_MODE(GENERAL);
;  1159          1280  3         BUILTIN
;  1160          1281  3             CALLG;
;  1161          1282  3         CALLG (.DUMMY_MESS_VECT, LIB$SIGNAL);
;  1162          1283  2         END;
;  1163          1284  2
;  1164          1285  2     ! Restore pointer field of PARSE_STG_DESC since this can be wiped out
;  1165          1286  2     ! during new style parsing.
;  1166          1287  2     !
;  1167          1288  2     IF .parse_stg_desc[dsc$a_pointer] EQL 0
;  1168          1289  2     THEN
;  1169          1290  2         parse_stg_desc[dsc$a_pointer] = .stg+.len;
;  1170          1291  2
;  1171          1292  2     ! finally, return a pointer to the verb node.
;  1172          1293  2     !
;  1173          1294  2     RETURN .VERB_NODE;
;  1174          1295  2
;  1175          1296  1     END; ! dbg$parse_allocate


                                                .EXTRN  LIB$SIGNAL

                                 007C 00000     .ENTRY  DBG$PARSE_ALLOCATE, Save R2,R3,R4,R5,R6      ; 1213
                              5E 04 C2 00002     SUBL2   #4, SP                                       ;
                              03 DD 00005     PUSHL   #3                                           ; 1243
                        FE19  CF 01 FB 00007     CALLS   #1, DBG$GET_TEMPMEM                          ;
                              56 50 D0 0000C     MOVL    R0, VERB_NODE                                ;
                              54 04 AC D0 0000F     MOVL    PARSE_STG_DESC, R4                        ; 1251
                              52 64 3C 00013     MOVZWL  (R4), LEN                                    ;
                              50 01 A2 9E 00016     MOVAB   1(R2), R0                                 ; 1252
                              50 04 C6 0001A     DIVL2   #4, R0                                       ;
                              01 A0 9F 0001D     PUSHAB  1(R0)                                        ;
                        FE00  CF 01 FB 00020     CALLS   #1, DBG$GET_TEMPMEM                          ;
                              53 50 D0 00025     MOVL    R0, STG                                      ;
                              55 04 A4 D0 00028     MOVL    4(R4), PARSE_STG_PTR                      ; 1253
                              50 01 CE 0002C     MNEGL   #1, I                                        ; 1259
                                 1A 11 0002F     BRB     3$                                           ;
                              51 85 90 00031  1$:    MOVB    (PARSE_STG_PTR)+, CHAR                    ; 1256
                           61 8F 51 91 00034     CMPB    CHAR, #97                                    ; 1257
                                 0D 1F 00038     BLSSU   2$                                           ;
                           7A 8F 51 91 0003A     CMPB    CHAR, #122                                   ;
                                 07 1A 0003E     BGTRU   2$                                           ;
                     6043        51 20 83 00040     SUBB3   #32, CHAR, (I)[STG]                       ; 1259
                                 04 11 00045     BRB     3$                                           ;
                           6043     51 90 00047  2$:    MOVB    CHAR, (I)[STG]                        ; 1261
                        E2       50 52 F2 0004B  3$:    AOBLSS  LEN, I, 1$                            ; 1254
                              6243  0D 90 0004F     MOVB    #13, (LEN)[STG]                           ; 1263
```

```
                        04    A4              53  D0 00053          MOVL    STG, 4(R4)                          ; 1264
                                              64  B6 00057          INCW    (R4)                                ; 1266
                                    4050      8F  BB 00059          PUSHR   #^M<R4,R6,SP>                       ; 1270
                      FEDD    CF              03  FB 0005D          CALLS   #3, DBG$NPARSE_ALLOCATE             ;
                              08              50  E8 00062          BLBS    R0, 4$                              ;
                00000000G     00        00    BE  FA 00065          CALLG   @DUMMY_MESS_VECT, LIB$SIGNAL        ; 1282
                              50        04    AC  D0 00069 4$:      MOVL    PARSE_STG_DESC, R0                  ; 1288
                                        04    A0  D5 00071          TSTL    4(R0)                               ;
                                              05  12 00074          BNEQ    5$                                  ;
                  04    A0                    53  C1 00076          ADDL3   LEN, STG, 4(R0)                     ; 1290
                              50              56  D0 0007B 5$:      MOVL    VERB_NODE, R0                       ; 1294
                                                  04 0007E          RET                                         ; 1296

; Routine Size:  127 bytes,    Routine Base:  DBG$CODE + 06A6
```

```
;  1177          1297  1 GLOBAL ROUTINE DBG$POP_TEMPMEM(POOLID): NOVALUE =
;  1178          1298  1 !
;  1179          1299  1 ! FUNCTION
;  1180          1300  1 !       This routine pops the pointers to the temporary memory blocks off
;  1181          1301  1 !       the Temporary Memory Pool Stacks.  Releases the temporary memory
;  1182          1302  1 !       blocks for each pointer.
;  1183          1303  1 !
;  1184          1304  1 ! INPUTS
;  1185          1305  1 !       POOLID - Stack ID to the Temporary Memory Pool Stacks, i.e., pops
;  1186          1306  1 !                from the current stack ID to POOLID.
;  1187          1307  1 !
;  1188          1308  1 ! OUTPUTS
;  1189          1309  1 !       None.
;  1190          1310  1 !
;  1191          1311  1
;  1192          1312  2     BEGIN
;  1193          1313  2
;  1194          1314  2     LOCAL
;  1195          1315  2         BLKPTR: REF VECTOR[,LONG],      ! Pointer to the current temporary
;  1196          1316  2                                         !       block to release
;  1197          1317  2         NEXTBLK;                        ! Pointer to the next block on the
;  1198          1318  2                                         !       chain
;  1199          1319  2
;  1200          1320  2
;  1201          1321  2
;  1202          1322  2     ! Make sure when we do the pop operation won't cause stack underflow.
;  1203          1323  2     !
;  1204          1324  3     IF (.POOLID LSS 1) OR (.POOLID GTR .DBG$TEMPMEM_POOLID)
;  1205          1325  2     THEN
;  1206          1326  2         $DBG_ERROR('GETMEMORY\DBG$POP_TEMPMEM stack underflow');
;  1207          1327  2
;  1208          1328  2
;  1209          1329  2     ! Pop from the current level to the given level.  For each pop operation,
;  1210          1330  2     ! release the temporary memory blocks pointed by DBG$TEMP_MEMORY.
;  1211          1331  2     !
;  1212          1332  2     BLKPTR = .DBG$TEMP_MEMORY;
;  1213          1333  2     DECR I FROM .DBG$TEMPMEM_POOLID TO .POOLID DO
;  1214          1334  3         BEGIN
;  1215          1335  3         WHILE .BLKPTR NEQ 0 DO
;  1216          1336  4             BEGIN
;  1217          1337  4             NEXTBLK = .BLKPTR[0];
;  1218          1338  4             DBG$REL_MEMORY(.BLKPTR);
;  1219          1339  4             BLKPTR = .NEXTBLK;
;  1220          1340  3             END;
;  1221          1341  3
;  1222          1342  3         BLKPTR = .DBG$TEMPMEM_POOLSTK[.I - 1];
;  1223          1343  2         END;
;  1224          1344  2
;  1225          1345  2
;  1226          1346  2     ! Adjust the current Temporary Memory Pool Stack pointer and the current
;  1227          1347  2     ! Pointer to the Temporary Memory blocks.
;  1228          1348  2     !
;  1229          1349  2     DBG$TEMP_MEMORY = .DBG$TEMPMEM_POOLSTK[.POOLID - 1];
;  1230          1350  2     DBG$TEMPMEM_POOLID = .POOLID - 1;
;  1231          1351  2     RETURN 0;
;  1232          1352  2
;  1233          1353  1     END;
```

```
                                                                 .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

24 47 42 44 5C 59 52 4F 4D 45 4D 54 45 47 29  00070 P.AAF:   .ASCII  \)GETMEMORY\<92>\DBG$POP_TEMPMEM stack u\   ;
61 74 73 20 4D 45 4D 50 4D 45 54 5F 50 4F 50  0007F                                                              ;
                                    75 20 6B 63  0008E                                                              ;
                        77 6F 6C 66 72 65 64 6E  00092          .ASCII  \nderflow\                                  ;


                                                                 .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                         003C 00000          .ENTRY  DBG$POP_TEMPMEM, Save R2,R3,R4,R5          ; 1297
                   55 00000000'  EF   9E 00002          MOVAB   DBG$TEMPMEM_POOLID, R5
                            04   AC   D5 00009          TSTL    POOLID                                     ; 1324
                            06   15      0000C          BLEQ    1$                                         ;
                   65       04   AC   D1 0000E          CMPL    POOLID, DBG$TEMPMEM_POOLID                 ;
                            15   15      00012          BLEQ    2$                                         ;
                      00000000'  EF   9F 00014 1$:      PUSHAB  P.AAF                                      ; 1326
                                 01   DD 0001A          PUSHL   #1                                         ;
                      00028362   8F   DD 0001C          PUSHL   #164706                                    ;
          00000000G   00         03   FB 00022          CALLS   #3, LIB$SIGNAL                             ;
                      53    FC   A5   D0 00029 2$:      MOVL    DBG$TEMP_MEMORY, BLKPTR                    ; 1332
                      52         65   D0 0002D          MOVL    DBG$TEMPMEM_POOLID, I                      ; 1333
                                 19   11 00030          BRB     5$                                         ;
                                 53   D5 00032 3$:      TSTL    BLKPTR                                     ; 1335
                                 0F   13 00034          BEQL    4$                                         ;
                      54         63   D0 00036          MOVL    (BLKPTR), NEXTBLK                          ; 1337
                                 53   DD 00039          PUSHL   BLKPTR                                     ; 1338
              0000V   CF         01   FB 0003B          CALLS   #1, DBG$REL_MEMORY                         ;
                      53         54   D0 00040          MOVL    NEXTBLK, BLKPTR                            ; 1339
                                 ED   11 00043          BRB     3$                                         ; 1335
                      53       6542   D0 00045 4$:      MOVL    DBG$TEMPMEM_POOLSTK-4[I], BLKPTR           ; 1342
                                 52   D7 00049          DECL    I                                          ; 1333
                 04   AC         52   D1 0004B 5$:      CMPL    I, POOLID                                  ;
                                 E1   18 0004F          BGEQ    3$                                         ;
                      50    04   AC   D0 00051          MOVL    POOLID, R0                                 ; 1349
                 FC   A5       6540   D0 00055          MOVL    DBG$TEMPMEM_POOLSTK-4[R0], DBG$TEMP_MEMORY ;
                      65    FF   A0   9E 0005A          MOVAB   -1(R0), DBG$TEMPMEM_POOLID                 ; 1350
                                      04 0005E          RET                                                ; 1353

; Routine Size:  95 bytes,    Routine Base:  DBG$CODE + 0725
```

```
;  1235          1354  1 GLOBAL ROUTINE DBG$PUSH_TEMPMEM =
;  1236          1355  1 !
;  1237          1356  1 ! FUNCTION
;  1238          1357  1 !       This routine pushes the current pointer to temporary memory blocks
;  1239          1358  1 !       on the Temporary Memory Pool Stack.  Reset the current pointer to 0,
;  1240          1359  1 !       so the new temporary memory blocks can be initiated.
;  1241          1360  1 !
;  1242          1361  1 ! INPUTS
;  1243          1362  1 !       None.
;  1244          1363  1 !
;  1245          1364  1 ! OUTPUTS
;  1246          1365  1 !       The current Stack ID is returned.
;  1247          1366  1 !
;  1248          1367  1
;  1249          1368  2     BEGIN
;  1250          1369  2
;  1251          1370  2
;  1252          1371  2     ! Increment the Temporary Memory Pool Stack pointer.
;  1253          1372  2     !
;  1254          1373  2     DBG$TEMPMEM_POOLID = .DBG$TEMPMEM_POOLID + 1;
;  1255          1374  2
;  1256          1375  2
;  1257          1376  2     ! Make sure the stack won't overflow by above operation.
;  1258          1377  2     !
;  1259          1378  2     IF .DBG$TEMPMEM_POOLID GTR 25
;  1260          1379  2     THEN
;  1261          1380  2         $DBG_ERROR('GETMEMORY\DBG$PUSH_TEMPMEM stack overflow');
;  1262          1381  2
;  1263          1382  2
;  1264          1383  2     ! Save the current temporary memory block pointer on the stack, and
;  1265          1384  2     ! initiate a new temporary memory block pointer.
;  1266          1385  2     !
;  1267          1386  2     DBG$TEMPMEM_POOLSTK[.DBG$TEMPMEM_POOLID - 1] = .DBG$TEMP_MEMORY;
;  1268          1387  2     DBG$TEMP_MEMORY = 0;
;  1269          1388  2     RETURN .DBG$TEMPMEM_POOLID;
;  1270          1389  1     END;


                                                        .PSECT  DBG$PLIT,NOWRT,  SHR,  PIC,0

24 47 42 44 5C 59 52 4F 4D 45 4D 54 45 47 29  0009A P.AAG:  .ASCII  \)GETMEMORY\<92>\DBG$PUSH_TEMPMEM stack \   ;
74 73 20 4D 45 4D 50 4D 45 54 5F 48 53 55 50  000A9                                                           ;
                                  20 6F 63 61  000B8                                                           ;
                        77 6F 6C 66 72 65 76 6F 000BC           .ASCII  \overflow\                            ;


                                                        .PSECT  DBG$CODE,NOWRT,  SHR,  PIC,0

                                              0004 00000        .ENTRY  DBG$PUSH_TEMPMEM, Save R2                ; 1354
                                52 00000000'  EF  9E 00002      MOVAB   DBG$TEMPMEM_POOLID, R2                   ;
                                              62  D6 00009      INCL    DBG$TEMPMEM_POOLID                       ; 1373
                                19            62  D1 0000B      CMPL    DBG$TEMPMEM_POOLID, #25                  ; 1378
                                              15  15 0000E      BLEQ    1$                                       ;
                                   00000000'  EF  9F 00010      PUSHAB  P.AAG                                    ; 1380
                                              01  DD 00016      PUSHL   #1                                       ;
```

```
                00028362   8F  DD 00018          PUSHL   #164706                                                          ;
      00000000G   00       03  FB 0001E          CALLS   #3, LIB$SIGNAL                                                  ;
                  50       62  D0 00025 1$:      MOVL    DBG$TEMPMEM_POOLID, R0                                          ; 1386
                6240    FC  A2  D0 00028          MOVL    DBG$TEMP_MEMORY, DBG$TEMPMEM_POOLSTK-4[R0]                      ;
                    FC  A2  D4 0002D              CLRL    DBG$TEMP_MEMORY                                                 ; 1387
                  50        62  D0 00030          MOVL    DBG$TEMPMEM_POOLID, R0                                          ; 1388
                            04 00033              RET                                                                     ; 1389

; Routine Size:  52 bytes,    Routine Base:  DBG$CODE + 0784
```

```
;  1272          1390  1 GLOBAL ROUTINE DBG$REL_MEMORY(ADDRESS): NOVALUE =
;  1273          1391  1 !
;  1274          1392  1 ! FUNCTION
;  1275          1393  1 !       This routine releases the memory block at a specified address to the
;  1276          1394  1 !       free memory pool.  The memory block is coalesced with the previous and
;  1277          1395  1 !       the following memory blocks if these are free and is added to the free-
;  1278          1396  1 !       list.
;  1279          1397  1 !
;  1280          1398  1 ! INPUTS
;  1281          1399  1 !       ADDRESS - The address of the memory block to be released.  This must be
;  1282          1400  1 !                 the same address as originally returned by DBG$GET_MEMORY when
;  1283          1401  1 !                 the block was allocated.
;  1284          1402  1 !
;  1285          1403  1 ! OUTPUTS
;  1286          1404  1 !       The specified memory block is released.  No value is returned.
;  1287          1405  1 !
;  1288          1406  1
;  1289          1407  2     BEGIN
;  1290          1408  2
;  1291          1409  2     MAP
;  1292          1410  2         ADDRESS: REF FMEM$BLOCK;        ! Pointer to the allocated part of the
;  1293          1411  2                                         !      block to be released
;  1294          1412  2
;  1295          1413  2     LOCAL
;  1296          1414  2         BACKPTR: REF FMEM$BLOCK,        ! Pointer to previous free-list block
;  1297          1415  2         BLKPTR:  REF FMEM$BLOCK,        ! Pointer to the released memory block
;  1298          1416  2         FORWPTR: REF FMEM$BLOCK,        ! Pointer to the next free-list block
;  1299          1417  2         NEXTBLK: REF FMEM$BLOCK,        ! Pointer to the next sequential memory
;  1300          1418  2                                         !      block in the memory pool
;  1301          1419  2         PREVBLK: REF FMEM$BLOCK;        ! Pointer to the previous sequential
;  1302          1420  2                                         !      memory block in the memory pool
;  1303          1421  2
;  1304          1422  2
;  1305          1423  2
;  1306          1424  2     ! Pick up the address of the memory block header.  Then do some integrity
;  1307          1425  2     ! checks to make sure this is a valid allocated memory block.
;  1308          1426  2     !
;  1309          1427  2     BLKPTR = ADDRESS[FMEM$A_HEADER];
;  1310          1428  2     IF (.BLKPTR[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL) OR
;  1311          1429  3        (NOT .BLKPTR[FMEM$V_THISALLOC])
;  1312          1430  2     THEN
;  1313          1431  2         SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  1314          1432  2
;  1315          1433  2
;  1316          1434  2     ! See if we can combine the released block with the previous sequential
;  1317          1435  2     ! block in the memory pool.  If we can, we make BLKPTR point to that prev-
;  1318          1436  2     ! ious block and increase its length accordingly.
;  1319          1437  2     !
;  1320          1438  2     IF NOT .BLKPTR[FMEM$V_PREVALLOC]
;  1321          1439  2     THEN
;  1322          1440  3         BEGIN
;  1323          1441  3         PREVBLK = .BLKPTR - 4*.BLKPTR[FMEM$L_PREVLEN];
;  1324          1442  3         IF .PREVBLK[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL
;  1325          1443  3         THEN
;  1326          1444  3             SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  1327          1445  3
;  1328          1446  3         PREVBLK[FMEM$V_LENGTH] =
```

```
;  1329          1447  3                 .PREVBLK[FMEM$V_LENGTH] + .BLKPTR[FMEM$V_LENGTH];
;  1330          1448  3          BLKPTR = .PREVBLK;
;  1331          1449  3          END
;  1332          1450  3
;  1333          1451  3
;  1334          1452  3      ! If the previous block was not free, mark the released block itself as
;  1335          1453  3      ! unallocated and link it into the free-list.
;  1336          1454  3      !
;  1337          1455  2      ELSE
;  1338          1456  3          BEGIN
;  1339          1457  3          BLKPTR[FMEM$V_THISALLOC] = FALSE;
;  1340          1458  3          FORWPTR = .DBG$FREE_LIST[FMEM$L_FLINK];
;  1341          1459  3          BACKPTR = .DBG$FREE_LIST;
;  1342          1460  3          BLKPTR[FMEM$L_FLINK] = .FORWPTR;
;  1343          1461  3          BLKPTR[FMEM$L_BLINK] = .BACKPTR;
;  1344          1462  3          FORWPTR[FMEM$L_BLINK] = .BLKPTR;
;  1345          1463  3          BACKPTR[FMEM$L_FLINK] = .BLKPTR;
;  1346          1464  2          END;
;  1347          1465  2
;  1348          1466  2
;  1349          1467  2      ! Now see if the next sequential block in the memory pool is free.  If it
;  1350          1468  2      ! is, we unlink it from the free-list and coalesce it with the BLKPTR block.
;  1351          1469  2      !
;  1352          1470  2      NEXTBLK = .BLKPTR + 4*.BLKPTR[FMEM$V_LENGTH];
;  1353          1471  2      IF .NEXTBLK[FMEM$B_SENTINEL] NEQ FMEM$K_SENTINEL
;  1354          1472  2      THEN
;  1355          1473  2          SIGNAL(DBG$_INTMEMERR, 1, .BLKPTR);
;  1356          1474  2
;  1357          1475  2      IF NOT .NEXTBLK[FMEM$V_THISALLOC]
;  1358          1476  2      THEN
;  1359          1477  3          BEGIN
;  1360          1478  3          FORWPTR = .NEXTBLK[FMEM$L_FLINK];
;  1361          1479  3          BACKPTR = .NEXTBLK[FMEM$L_BLINK];
;  1362          1480  3          FORWPTR[FMEM$L_BLINK] = .BACKPTR;
;  1363          1481  3          BACKPTR[FMEM$L_FLINK] = .FORWPTR;
;  1364          1482  3          BLKPTR[FMEM$V_LENGTH] =
;  1365          1483  3                  .BLKPTR[FMEM$V_LENGTH] + .NEXTBLK[FMEM$V_LENGTH];
;  1366          1484  2          END;
;  1367          1485  2
;  1368          1486  2
;  1369          1487  2      ! We have now done all coalescing we can do.  Set the length of the block at
;  1370          1488  2      ! the end of the block and mark it there as being unallocated.  Then return.
;  1371          1489  2      !
;  1372          1490  2      NEXTBLK = .BLKPTR + 4*.BLKPTR[FMEM$V_LENGTH];
;  1373          1491  2      NEXTBLK[FMEM$L_PREVLEN] = .BLKPTR[FMEM$V_LENGTH];
;  1374          1492  2      NEXTBLK[FMEM$V_PREVALLOC] = FALSE;
;  1375          1493  2      RETURN;
;  1376          1494  2
;  1377          1495  1      END;


                                  007C 00000          .ENTRY  DBG$REL_MEMORY, Save R2,R3,R4,R5,R6          ; 1390
                     56 00000000G   00   9E 00002     MOVAB   LIB$SIGNAL, R6                               ;
           53        04   AC             04   C3 00009     SUBL3   #4, ADDRESS, BLKPTR                          ; 1427
```

```
                        B2  8F      03      A3  91 0000E          CMPB    3(BLKPTR), #178                 ;1428
                                            04  12 00013          BNEQ    1$
                    0D          63          16  E0 00015          BBS     #22, (BLKPTR), 2$               ;1429
                                            53  DD 00019 1$:      PUSHL   BLKPTR                          ;1431
                                            01  DD 0001B          PUSHL   #1
                            00028AF4        8F  DD 0001D          PUSHL   #166644
                                66          03  FB 00023          CALLS   #3, LIB$SIGNAL
                    36          63          17  E0 00026 2$:      BBS     #23, @LKPTR), 4$                ;1438
                                50      FC  A3  D0 0002A          MOVL    -4(BLKPTR), R0                  ;1441
                                50          04  C4 0002E          MULL2   #4, R0
                    52          53          50  C3 00031          SUBL3   R0, BLKPTR, PREVBLK
                        B2  8F      03      A2  91 00035          CMPB    3(PREVBLK), #178                ;1442
                                            0D  13 0003A          BEQL    3$
                                            53  DD 0003C          PUSHL   BLKPTR                          ;1444
                                            01  DD 0003E          PUSHL   #1
                            00028AF4        8F  DD 00040          PUSHL   #166644
                                66          03  FB 00046          CALLS   #3, LIB$SIGNAL
        50          62          16          00  EF 00049 3$:      EXTZV   #0, #22, (PREVBLK), R0          ;1447
        51          63          16          00  EF 0004E          EXTZV   #0, #22, (BLKPTR), R1
                                50          51  C0 00053          ADDL2   R1, R0
        62          16          00          50  F0 00056          INSV    R0, #0, #22, (PREVBLK)
                                53          52  D0 0005B          MOVL    PREVBLK, BLKPTR                 ;1448
                                            1F  11 0005E          BRB     5$                              ;1438
                        02  A3      40      8F  8A 00060 4$:      BICB2   #64, 2(BLKPTR)                  ;1457
                            50 00000000'    EF  D0 00065          MOVL    DBG$FREE_LIST, R0               ;1458
                                54      04  A0  D0 0006C          MOVL    4(R0), FORWPTR
                                55          50  D0 00070          MOVL    R0, BACKPTR                     ;1459
                        04  A3              54  7D 00073          MOVQ    FORWPTR, 4(BLKPTR)              ;1460
                        08  A4              53  D0 00077          MOVL    BLKPTR, 8(FORWPTR)              ;1462
                        04  A5              53  D0 0007B          MOVL    BLKPTR, 4(BACKPTR)              ;1463
        50          63          16          00  EF 0007F 5$:      EXTZV   #0, #22, (BLKPTR), R0           ;1470
                                52      6340    DE 00084          MOVAL   (BLKPTR)[R0], NEXTBLK
                        B2  8F      03      A2  91 00088          CMPB    3(NEXTBLK), #178                ;1471
                                            0D  13 0008D          BEQL    6$
                                            53  DD 0008F          PUSHL   BLKPTR                          ;1473
                                            01  DD 00091          PUSHL   #1
                            00028AF4        8F  DD 00093          PUSHL   #166644
                                66          03  FB 00099          CALLS   #3, LIB$SIGNAL
                    1E          62          16  E0 0009C 6$:      BBS     #22, (NEXTBLK), 7$              ;1475
                                54      04  A2  7D 000A0          MOVQ    4(NEXTBLK), FORWPTR             ;1478
                        08  A4              55  D0 000A4          MOVL    BACKPTR, 8(FORWPTR)             ;1480
                        04  A5              54  D0 000A8          MOVL    FORWPTR, 4(BACKPTR)             ;1481
        50          63          16          00  EF 000AC          EXTZV   #0, #22, (BLKPTR), R0           ;1483
        51          62          16          00  EF 000B1          EXTZV   #0, #22, (NEXTBLK), R1
                                50          51  C0 000B6          ADDL2   R1, R0
        63          16          00          50  F0 000B9          INSV    R0, #0, #22, (BLKPTR)
        50          63          16          00  EF 000BE 7$:      EXTZV   #0, #22, (BLKPTR), R0           ;1490
                                52      6340    DE 000C3          MOVAL   (BLKPTR)[R0], NEXTBLK
                        FC  A2              50  D0 000C7          MOVL    R0, -4(NEXTBLK)                 ;1491
                        02  A2      80      8F  8A 000CB          BICB2   #128, 2(NEXTBLK)                ;1492
                                                04 000D0          RET                                     ;1495

; Routine Size:  209 bytes,    Routine Base:  DBG$CODE + 07B8
```

```
;1379     1496  1 GLOBAL ROUTINE DBG$REL_TEMPMEM: NOVALUE =
;1380     1497  1 !
;1381     1498  1 ! FUNCTION
;1382     1499  1 !       This routine releases all "temporary" memory blocks allocated by routine
;1383     1500  1 !       DBG$GET_TEMPMEM.  This routine is normally called after the completion
;1384     1501  1 !       of each command--it thus cleans up any storage used in processing the
;1385     1502  1 !       command without requiring an explicit release call for each such block.
;1386     1503  1 !
;1387     1504  1 ! INPUTS
;1388     1505  1 !       NONE
;1389     1506  1 !
;1390     1507  1 ! OUTPUTS
;1391     1508  1 !       All "temporary" blocks on the DBG$TEMP_MEMORY list are released to the
;1392     1509  1 !               memory pool.  No value is returned.
;1393     1510  1 !
;1394     1511  1
;1395     1512  2     BEGIN
;1396     1513  2
;1397     1514  2     LOCAL
;1398     1515  2         BLKPTR: REF VECTOR[,LONG],      ! Pointer to the current temporary block
;1399     1516  2                                         !       to release
;1400     1517  2         NEXTBLK;                        ! Pointer to the next block on the chain
;1401     1518  2
;1402     1519  2
;1403     1520  2
;1404     1521  2     ! If this is a Testable Debugger, check the condition of the memory pool.
;1405     1522  2     !
;1406     1523  2     IF .DBG$GV_CONTROL[DBG$V_CONTROL_TDBG] THEN DBG$CHECK_MEMORY();
;1407     1524  2
;1408     1525  2
;1409     1526  2     ! Clear the RST Reference List to contain zero entries.  This says that the
;1410     1527  2     ! current Debug command has ended, and the RST entries it referenced are no
;1411     1528  2     ! longer being referenced by SYMIDs elsewhere in the Debugger.
;1412     1529  2     !
;1413     1530  2     RST$REF_LIST[1] = 0;
;1414     1531  2
;1415     1532  2
;1416     1533  2     ! Loop through the singly linked list pointed to by DBG$TEMP_MEMORY and
;1417     1534  2     ! release each block on the list to the free memory pool.
;1418     1535  2     !
;1419     1536  2     BLKPTR = .DBG$TEMP_MEMORY;
;1420     1537  2     WHILE TRUE DO
;1421     1538  3         BEGIN
;1422     1539  3         WHILE .BLKPTR NEQ 0 DO
;1423     1540  4             BEGIN
;1424     1541  4             NEXTBLK = .BLKPTR[0];
;1425     1542  4             DBG$REL_MEMORY(.BLKPTR);
;1426     1543  4             BLKPTR = .NEXTBLK;
;1427     1544  3             END;
;1428     1545  3
;1429     1546  3         IF .DBG$TEMPMEM_POOLID EQL 0 THEN EXITLOOP;
;1430     1547  3         DBG$TEMP_MEMORY = .DBG$TEMPMEM_POOLSTK[.DBG$TEMPMEM_POOLID - 1];
;1431     1548  3         BLKPTR = .DBG$TEMP_MEMORY;
;1432     1549  3         DBG$TEMPMEM_POOLID = .DBG$TEMPMEM_POOLID - 1;
;1433     1550  2         END;
;1434     1551  2
;1435     1552  2     DBG$TEMP_MEMORY = 0;
```

```
; 1436          1553  2     RETURN;
; 1437          1554  2
; 1438          1555  1     END;


                                    001C 00000          .ENTRY  DBG$REL_TEMPMEM, Save R2,R3,R4                  ; 1496
                54 00000000'    EF   9E 00002          MOVAB   DBG$TEMP_MEMORY, R4                             ;
                05 00000000G    00   E9 00009          BLBC    DBG$GV_CONTROL, 1$                              ; 1523
            F762                CF   FB 00010          CALLS   #0, DBG$CHECK_MEMORY                            ;
                50 00000000G    00   D0 00015 1$:      MOVL    RSI$REF_LIST, R0                                ; 1530
                         04     A0   D4 0001C          CLRL    4(R0)                                           ;
                52              64   D0 0001F          MOVL    DBG$TEMP_MEMORY, BLKPTR                         ; 1536
                                52   D5 00022 2$:      TSTL    BLKPTR                                          ; 1539
                                0F   13 00024          BEQL    3$                                              ;
                53              62   D0 00026          MOVL    (BLKPTR), NEXTBLK                               ; 1541
                                52   DD 00029          PUSHL   BLKPTR                                          ; 1542
            FEFF                CF   01 FB 0002B       CALLS   #1, DBG$REL_MEMORY                              ;
                52              53   D0 00030          MOVL    NEXTBLK, BLKPTR                                 ; 1543
                                ED   11 00033          BRB     2$                                              ; 1539
                50        04    A4   D0 00035 3$:      MOVL    DBG$TEMPMEM_POOLID, R0                          ; 1546
                                0D   13 00039          BEQL    4$                                              ;
                64        04 A440   D0 0003B          MOVL    DBG$TEMPMEM_POOLSTK-4[R0], DBG$TEMP_MEMORY      ; 1547
                52              64   D0 00040          MOVL    DBG$TEMP_MEMORY, BLKPTR                         ; 1548
                          04    A4   D7 00043          DECL    DBG$TEMPMEM_POOLID                              ; 1549
                                DA   11 00046          BRB     2$                                              ; 1537
                                64   D4 00048 4$:      CLRL    DBG$TEMP_MEMORY                                 ; 1552
                                     04 0004A          RET                                                     ; 1555

; Routine Size:  75 bytes,    Routine Base:  DBG$CODE + 0889

; 1439          1556  1
; 1440          1557  0 END ELUDOM


                                                       .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
;  DBG$OWN                       116  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                     2260  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$PLIT                      196  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)


;                            Library Statistics
;
;                                 -------- Symbols --------      Pages       Processing
;      File                       Total    Loaded   Percent      Mapped      Time
```

```
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                18619         9         0      1000          00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1           32         0         0         7          00:00.2
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1           1545        45         2        97          00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                                 418         0         0        31          00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1            386         7         1        22          00:00.3
```

```
;                                    COMMAND QUALIFIERS

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

; Size:          2260 code + 312 data bytes
; Run Time:         00:42.2
; Elapsed Time:     00:46.6
; Lines/CPU Min:     2214
; Lexemes/CPU-Min:  15433
; Memory Used:  200 pages
; Compilation Complete
```

GETMEMORY
LIS

DSTRECORDS
LIS

ISSH
LIS

RESETSSI
LIS