F 1

```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
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  EEE              BBB         BBB  UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     BBBBBBBBBBBB     UUU         UUU  GGG
DDD         DDD  EEEEEEEEEEEE     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     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUU     GGGGGGGGG
```

```
                                              G  6
**FILE**ID**DBGNERMSG










DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  EEEEEEEEEE  RRRRRRRR    MM      MM    SSSSSSSS    GGGGGGGG
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  EEEEEEEEEE  RRRRRRRR    MM      MM    SSSSSSSS    GGGGGGGG
DD      DD  BB      BB  GG          NN      NN  EE          RR      RR  MMMM  MMMM  SS          GG
DD      DD  BB      BB  GG          NN      NN  EE          RR      RR  MMMM  MMMM  SS          GG
DD      DD  BB      BB  GG          NNNN    NN  EE          RR      RR  MM  MM  MM  SS          GG
DD      DD  BB      BB  GG          NNNN    NN  EE          RR      RR  MM  MM  MM  SS          GG
DD      DD  BBBBBBBB    GG          NN  NN  NN  EEEEEEEE    RRRRRRRR    MM      MM    SSSSSS    GG
DD      DD  BBBBBBBB    GG          NN  NN  NN  EEEEEEEE    RRRRRRRR    MM      MM    SSSSSS    GG
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  EE          RR  RR      MM      MM          SS  GG  GGGGGG
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  EE          RR  RR      MM      MM          SS  GG  GGGGGG
DD      DD  BB      BB  GG      GG  NN      NN  EE          RR    RR    MM      MM          SS  GG      GG      ....
DD      DD  BB      BB  GG      GG  NN      NN  EE          RR    RR    MM      MM          SS  GG      GG      ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  EEEEEEEEEE  RR      RR  MM      MM  SSSSSSSS      GGGGGG        ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  EEEEEEEEEE  RR      RR  MM      MM  SSSSSSSS      GGGGGG        ....


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
```

```
0 MODULE DBGNERMSG (IDENT = 'V04-000') =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:     DEBUG
1 !
1 ! ABSTRACT:
1 !
1 !       Version 3 debugger error output routines are contained in this module. In
1 !       contrast to the version 2 debugger, error messages are not handled by the
1 !       exception handling mechanism. That is, error messages are not SIGNALed. The
1 !       routines in this module call SYS$PUTMSG to recover and format the DEBUG
1 !       messages from the system message file. The address of the version 2 debugger
1 !       routine dbg$out_message is suppied as an action routine. It is this routine
1 !       which actually outputs the message. SYS$PUTMSG is used instead of SYS$GETMSG
1 !       because the parameters to SYS$PUTMSG resemble the the vector of longwords
1 !       formed by a SIGNAL, a format which dbg$out_message expects.
1 !
1 ! ENVIRONMENT:  VAX/VMS
1 !
1 ! AUTHOR:       David Plummer, CREATION DATE:   4/10/80
1 !
1 ! MODIFIED BY:
1 !               David Plummer, 10-Jul-80, DLP
1 !
1 !
1 ! 2.2-001       10-Jul-80       DLP     Added check for a null message vector ptr
1 !
1 !
1 !       R. Title        Feb 1983        Added parse and execute of DUMP
1 !                                       command to this module (for lack
1 !                                       of a better place to put it).
```

```
;   58          0058  1 !                                          This command is used by developers
;   59          0059  1 !                                          to dump DEBUG internals.
;   60          0060  1
;   61          0061  1 ! VERSION:        V02.2-002
;   62          0062  1 !
;   63          0063  1 !--
;    65         0064  1 !
;    66         0065  1 ! TABLE OF CONTENTS:
;    67         0066  1 !
;    68         0067  1
;    69         0068  1 FORWARD ROUTINE
;    70         0069  1         DBG$NPARSE_DUMP,                        ! Parse DUMP command
;    71         0070  1         DBG$NEXECUTE_DUMP,                      ! Execute DUMP command
;    72         0071  1         DBG$NOUT_INFO,                          ! Outputs an informational message
;    73         0072  1         DBG$NMAKE_ARG_VECT,                     ! Constructs a standard message argument vector
;    74         0073  1         DBG$NOUT_ARG_VECT : NOVALUE,            ! Outputs a message argument vector
;    75         0074  1         DBG$NSYNTAX_ERROR;                      ! Constructs a syntax error message vect
;    76         0075  1
;    77         0076  1 !
;    78         0077  1 ! REQUIRE FILES:
;    79         0078  1 !
;    80         0079  1
;    81         0080  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    82         0214  1
;    83         0215  1 !
;    84         0216  1 ! EXTERNAL REFERENCES:
;    85         0217  1 !
;    86         0218  1 EXTERNAL ROUTINE
;    87         0219  1         DBG$ANALYZE_HASH: NOVALUE,                              ! Dump info about hash chains
;    88         0220  1         DBG$DUMP_GLOBAL: NOVALUE,                               ! Dump info about GST
;    89         0221  1         DBG$DUMP_SAT: NOVALUE,                                  ! Dump info about SAT
;    90         0222  1         DBG$GET_TEMPMEM,                                        ! Allocates listed dynamic storage
;    91         0223  1         DBG$NMATCH,                                             ! Match input string
;    92         0224  1         DBG$NNEXT_WORD,                                         ! Get next word in input
;    93         0225  1         DBG$OUT_MESSAGE : NOVALUE,                              ! Called as an action routine by SYS$PUTMSG to
;    94         0226  1                                                                 ! output the error message.
;    95         0227  1         SYS$PUTMSG      : ADDRESSING_MODE (GENERAL);            ! System message output routine
;    96         0228  1
;    97         0229  1 EXTERNAL
;    98         0230  1         DBG$GL_DEVELOPER: BITVECTOR[];                          ! Developer flags
;    99         0231  1
;   100         0232  1 !
;   101         0233  1 ! LITERALS
;   102         0234  1 !
;   103         0235  1 ! Used for communication between PARSE_DUMP and EXECUTE_DUMP.
;   104         0236  1 !
;   105         0237  1 LITERAL
;   106         0238  1         DUMP_MIN  = 0,
;   107         0239  1         DUMP_HASH = 0,
;   108         0240  1         DUMP_SAT  = 1,
;   109         0241  1         DUMP_GST  = 2,
;   110         0242  1         DUMP_MAX  = 2;
```

```
GLOBAL ROUTINE DBG$NPARSE_DUMP (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
!
! FUNCTION
!       This routine parses the DUMP command. This command dumps internal
!       DEBUG data structures. The command is only available to developers.
!
! INPUTS
!       INPUT_DESC      - The remaining command string.
!       VERB_NODE       - Pointer to partially constructed parse tree
!       MESSAGE_VECT    - Error message vector
!
! OUTPUTS
!       Information is printed at the terminal.
!       The input string is updated to point beyond what we picked up.
!       A return status is returned.
!
    BEGIN
    MAP
        INPUT_DESC: REF DBG$STG_DESC,
        VERB_NODE: REF DBG$VERB_NODE;

    BIND
        DBG$CS_CR        = UPLIT BYTE (1, DBG$K_CAR_RETURN),
        DBG$CS_GST       = UPLIT BYTE (3, 'GST'),
        DBG$CS_HASH      = UPLIT BYTE (4, 'HASH'),
        DBG$CS_SAT       = UPLIT BYTE (3, 'SAT');

    ! Check developer flag 0. This enables the DUMP command.
    !
    IF NOT .DBG$GL_DEVELOPER[0]
    THEN
        BEGIN
        .MESSAGE_VECT = DBG$NSYNTAX_ERROR(UPLIT BYTE(%ASCIC 'DUMP'));
        RETURN STS$K_SEVERE;
        END;

    ! Pick up the keyword. At the moment, we only support DUMP HASH,
    ! but more keywords may be added later.
    !
    SELECTONE TRUE OF
        SET

        ! DUMP GST
        !
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_GST, 1)]:
            BEGIN
            VERB_NODE[DBG$L_VERB_OBJECT_PTR] = DUMP_GST;
            END;

        ! DUMP HASH.
        !
        [DBG$NMATCH (.INPUT_DESC, DBG$CS_HASH, 1)]:
            BEGIN
            VERB_NODE[DBG$L_VERB_OBJECT_PTR] = DUMP_HASH;
            END;

        ! DUMP SAT.
```

```
;  169       0300  2            !
;  170       0301  2            [DBG$NMATCH (.INPUT_DESC, DBG$CS_SAT, 1)]:
;  171       0302  3                BEGIN
;  172       0303  3                VERB_NODE[DBG$L_VERB_OBJECT_PTR] = DUMP_SAT;
;  173       0304  2                END;
;  174       0305  2
;  175       0306  2            ! Any other DUMP argument is a syntax error.
;  176       0307  2            !
;  177       0308  2            [OTHERWISE]:
;  178       0309  3                BEGIN
;  179       0310  4                .MESSAGE_VECT = (
;  180       0311  4                    IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1)
;  181       0312  4                    THEN
;  182       0313  4                        DBG$NMAKE_ARG_VECT(DBG$_NEEDMORE)
;  183       0314  4                    ELSE
;  184       0315  3                        DBG$NSYNTAX_ERROR(DBG$NNEXT_WORD(.INPUT_DESC)));
;  185       0316  3                RETURN STS$K_SEVERE;
;  186       0317  2                END;
;  187       0318  2            TES;
;  188       0319  2        RETURN STS$K_SUCCESS;
;  189       0320  1        END;


                                                       .TITLE  DBGNERMSG
                                                       .IDENT  \V04-000\

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

                           0D  01  00000 P.AAA:  .BYTE   1, 13                                       ;
                               03  00002 P.AAB:  .BYTE   3                                           ;
                       54  53  47  00003         .ASCII  \GST\                                       ;
                               04  00006 P.AAC:  .BYTE   4                                           ;
                   48  53  41  48  00007         .ASCII  \HASH\                                      ;
                               03  0000B P.AAD:  .BYTE   3                                           ;
                       54  41  53  0000C         .ASCII  \SAT\                                       ;
               50  4D  55  44  04  0000F P.AAE:  .ASCII  <4>\DUMP\                                   ;

                                         DBG$CS_CR=          P.AAA
                                         DBG$CS_GST=         P.AAB
                                         DBG$CS_HASH=        P.AAC
                                         DBG$CS_SAT=         P.AAD
                                                 .EXTRN  DBG$ANALYZE_HASH
                                                 .EXTRN  DBG$DUMP_GLOBAL
                                                 .EXTRN  DBG$DUMP_SAT, DBG$GET_TEMPMEM
                                                 .EXTRN  DBG$NMATCH, DBG$NNEXT_WORD
                                                 .EXTRN  DBG$OUT_MESSAGE
                                                 .EXTRN  SYS$PUTMSG, DBG$GL_DEVELOPER

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

                             001C  00000         .ENTRY  DBG$NPARSE_DUMP, Save R2,R3,R4      ; 0243
               54 00000000G  00  9E  00002         MOVAB   DBG$NMATCH, R4                      ;
               53 00000000'  EF  9E  00009         MOVAB   P.AAE, R3                           ;
               04 00000000G  00  E8  00010         BLBS    DBG$GL_DEVELOPER, 1$                ; 0272
                             53  DD  00017         PUSHL   R3                                  ; 0275
                             73  11  00019         BRB     6$                                  ;
                             01  DD  0001B 1$:     PUSHL   #1                                  ; 0287
```

```
                          F3    A3   9F 0001D        PUSHAB  DBG$CS_GST                          ;
             52           04    AC   D0 00020        MOVL    INPUT_DESC, R2                      ;
                                52   DD 00024        PUSHL   R2                                  ;
             64                 03   FB 00026        CALLS   #3, DBG$NMATCH                      ;
             01                 50   D1 00029        CMPL    R0, #1                              ;
                                0A   12 0002C        BNEQ    2$                                  ;
             50           08    AC   D0 0002E        MOVL    VERB_NODE, R0                       ; 0289
        08   A0                 02   D0 00032        MOVL    #2, 8(R0)                           ;
                                63   11 00036        BRB     8$                                  ; 0282
                                01   DD 00038 2$:    PUSHL   #1                                  ; 0294
                          F7    A3   9F 0003A        PUSHAB  DBG$CS_HASH                         ;
                                52   DD 0003D        PUSHL   R2                                  ;
             64                 03   FB 0003F        CALLS   #3, DBG$NMATCH                      ;
             01                 50   D1 00042        CMPL    R0, #1                              ;
                                09   12 00045        BNEQ    3$                                  ;
             50           08    AC   D0 00047        MOVL    VERB_NODE, R0                       ; 0296
                          08    A0   D4 0004B        CLRL    8(R0)                               ;
                                4B   11 0004E        BRB     8$                                  ; 0282
                                01   DD 00050 3$:    PUSHL   #1                                  ; 0301
                          FC    A3   9F 00052        PUSHAB  DBG$CS_SAT                          ;
                                52   DD 00055        PUSHL   R2                                  ;
             64                 03   FB 00057        CALLS   #3, DBG$NMATCH                      ;
             01                 50   D1 0005A        CMPL    R0, #1                              ;
                                0A   12 0005D        BNEQ    4$                                  ;
             50           08    AC   D0 0005F        MOVL    VERB_NODE, R0                       ; 0303
        08   A0                 01   D0 00063        MOVL    #1, 8(R0)                           ;
                                32   11 00067        BRB     8$                                  ; 0282
                                01   DD 00069 4$:    PUSHL   #1                                  ; 0311
                          F1    A3   9F 0006B        PUSHAB  DBG$CS_CR                           ;
                                52   DD 0006E        PUSHL   R2                                  ;
             64                 03   FB 00070        CALLS   #3, DBG$NMATCH                      ;
             0D                 50   E9 00073        BLBC    R0, 5$                              ;
                    000280D0    8F   DD 00076        PUSHL   #164048                             ; 0313
     0000V   CF                 01   FB 0007C        CALLS   #1, DBG$NMAKE_ARG_VECT              ;
                                10   11 00081        BRB     7$                                  ;
                                52   DD 00083 5$:    PUSHL   R2                                  ; 0315
 00000000G   00                 01   FB 00085        CALLS   #1, DBG$NNEXT_WORD                  ;
                                50   DD 0008C        PUSHL   R0                                  ;
     0000V   CF                 01   FB 0008E 6$:    CALLS   #1, DBG$NSYNTAX_ERROR               ;
        0C   BC                 50   D0 00093 7$:    MOVL    R0, @MESSAGE_VECT                   ; 0310
             50                 04   D0 00097        MOVL    #4, R0                              ; 0316
                                     04 0009A        RET                                         ;
             50                 01   D0 0009B 8$:    MOVL    #1, R0                              ; 0319
                                     04 0009E        RET                                         ; 0320

; Routine Size:  159 bytes,    Routine Base:  DBG$CODE + 0000
;  191        0321   1  GLOBAL ROUTINE DBG$NEXECUTE_DUMP (VERB_NODE, MESSAGE_VECT) =
;  192        0322   1  !
;  193        0323   1  ! FUNCTION
;  194        0324   1  !       Performs the action associated with the DUMP command.
;  195        0325   1  !
;  196        0326   1  ! INPUTS
;  197        0327   1  !       VERB_NODE         - A pointer to the command tree
;  198        0328   1  !       MESSAGE_VECT      - Error message vector
;  199        0329   1  !
;  200        0330   1  ! OUTPUTS
;  201        0331   1  !       Information about internal DEBUG data structures will be printed
;  202        0332   1  !       at the terminal. A status code is returned.
;  203        0333   1  !
;  204        0334   2      BEGIN
;  205        0335   2      MAP
;  206        0336   2          VERB_NODE: REF DBG$VERB_NODE;
;  207        0337   2
;  208        0338   2      ! Case on the DUMP keyword. DUMP HASH is the only one we currently
;  209        0339   2      ! support.
;  210        0340   2      !
;  211        0341   2      CASE .VERB_NODE[DBG$L_VERB_OBJECT_PTR] FROM DUMP_MIN TO DUMP_MAX OF
;  212        0342   2          SET
;  213        0343   2
;  214        0344   2          [DUMP_GST]:
;  215        0345   2              DBG$DUMP_GLOBAL();
;  216        0346   2
;  217        0347   2          [DUMP_HASH]:
;  218        0348   2              DBG$ANALYZE_HASH();
;  219        0349   2
;  220        0350   2          [DUMP_SAT]:
;  221        0351   2              DBG$DUMP_SAT();
;  222        0352   2
;  223        0353   2          [INRANGE, OUTRANGE]:
;  224        0354   2              $DBG_ERROR('DBGNERMSG\DBG$NEXECUTE_DUMP');
;  225        0355   2
;  226        0356   2          TES;
;  227        0357   2      RETURN STS$K_SUCCESS;
;  228        0358   1      END;


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

24  47  42  44  5C  47  53  4D  52  45  4E  47  42  44  1B  00014 P.AAF:  .ASCII  <27>\DBGNERMSG\<92>\DBG$NEXECUTE_DUMP\       ;
        50  4D  55  44  5F  45  54  55  43  45  58  45  4E  00023                                                               ;


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

                                            0000  00000            .ENTRY  DBG$NEXECUTE_DUMP, Save nothing                ; 0321
                        50          04  AC    D0  00002            MOVL    VERB_NODE, R0                                  ; 0341
            02          00          08  A0    CF  00006            CASEL   8(R0), #0, #2                                  ;
          001D        002F            0026        0000B 1$:        .WORD   3$-1$,-                                        ;
                                                                           4$-1$,-                                        ;
                                                                           2$-1$                                          ;
                          00000000'   EF    9F  00011            PUSHAB  P.AAF                                          ; 0354
```

```
                                01  DD 00017           PUSHL   #1                                     ;
                    00028362    8F  DD 00019           PUSHL   #164706                                ;
      00000000G  00             03  FB 0001F           CALLS   #3, LIB$SIGNAL                         ;
                                19  11 00026           BRB     5$                                     ;
      00000000G  00             00  FB 00028 2$:       CALLS   #0, DBG$DUMP_GLOBAL                    ; 0345
                                10  11 0002F           BRB     5$                                     ;
      00000000G  00             00  FB 00031 3$:       CALLS   #0, DBG$ANALYZE_HASH                   ; 0348
                                07  11 00038           BRB     5$                                     ;
      00000000G  00             00  FB 0003A 4$:       CALLS   #0, DBG$DUMP_SAT                       ; 0351
                 50             01  D0 00041 5$:       MOVL    #1, R0                                 ; 0357
                                    04 00044           RET                                            ; 0358

; Routine Size:  69 bytes,    Routine Base:  DBG$CODE + 009F
```

```
1 GLOBAL ROUTINE DBG$NOUT_INFO (ERROR_CODE) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine outputs an informational message to the user's terminal and/or
1 !       log file.
1 !
1 !       This routine will not output message that do not have an informational
1 !       level of severity.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       error_code      - A longword containing an integer value corresponding
1 !                         to a DEBUG info message code
1 !
1 !       [fao_count]     - A longword containing the number of fao arguments supplied
1 !                         in conjunction with the first message code. This optional
1 !                         parameter MUST be supplied if ANY fao arguments are supplied.
1 !
1 !       [fao_first, ...] - A longword containing an fao argument to be incorperated
1 !                         into the info message text
1 !
1 !       [next_code, next_count, next_fao, ...]
1 !
1 !                       - The next message code, fao_count, fao_argument sequence.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       sts$k_success (1) - Success. Informational message output.
1 !
1 !       sts$k_severe  (4) - Failure. Message not an info and not output.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Outputs an informational message(s) to the user's terminal and/or log file.
1 !
1 !--
2    BEGIN
2
2    BUILTIN
2        ACTUALCOUNT,
2        ACTUALPARAMETER;
2
2    LOCAL
```

```
;   287        0416  2          NUM_ACTUALS,            ! Number of actual parameters
;   288        0417  2          I,                                ! Loop counter
;   289        0418  2          ERROR_VECT,                       ! Message vector
;   290        0419  2          ARG_VECT : REF VECTOR;  ! The message argument vector
;   291        0420  2
;   292        0421  2
;   293        0422  2      ! Make sure that the message code corresponds to an info
;   294        0423  2      !
;   295        0424  2      IF .error_code <0, 3, 0> NEQ sts$k_info
;   296        0425  2      THEN
;   297        0426  2          RETURN sts$k_severe;
;   298        0427  2
;   299        0428  2
;   300        0429  2      ! Make the argument vector
;   301        0430  2      !
;   302        0431  2      num_actuals = actualcount ();
;   303        0432  2
;   304        0433  2      arg_vect = dbg$get_tempmem(.num_actuals + 1);
;   305        0434  2      arg_vect [0] = .num_actuals;
;   306        0435  2
;   307        0436  2      INCR i FROM 1 TO .num_actuals
;   308        0437  2      DO
;   309        0438  2          arg_vect [.i] = actualparameter (.i);
;   310        0439  2
;   311        0440  2
;   312        0441  2      ! Output the message
;   313        0442  2      !
;   314        0443  2      dbg$nout_arg_vect (.arg_vect);
;   315        0444  2
;   316        0445  2      RETURN sts$k_success;
;   317        0446  2
;   318        0447  1      END;          ! End of dbg$nout_info



                                        0004 00000         .ENTRY  DBG$NOUT_INFO, Save R2                          ; 0359
   03       04   AC          03       00  ED 00002         CMPZV   #0, #3, ERROR_CODE, #3                          ; 0424
                                      04  13 00008         BEQL    1$                                              ;
                             50       04  D0 0000A         MOVL    #4, R0                                          ; 0426
                                          04 0000D         RET                                                     ;
                             52       6C  9A 0000E 1$:     MOVZBL  (AP), NUM_ACTUALS                               ; 0431
                                  01  A2  9F 00011         PUSHAB  1(NUM_ACTUALS)                                  ; 0433
                   00000000G 00       01  FB 00014         CALLS   #1, DBG$GET_TEMPMEM                             ;
                             60       52  D0 0001B         MOVL    NUM_ACTUALS, (ARG_VECT)                         ; 0434
                                      51  D4 0001E         CLRL    I                                               ; 0436
                                      05  11 00020         BRB     3$                                              ;
                           6041     6C41  D0 00022 2$:     MOVL    (AP)[I], (ARG_VECT)[I]                          ; 0438
                 F7          51       52  F3 00027 3$:     AOBLEQ  NUM_ACTUALS, I, 2$                              ;
                                      50  DD 0002B         PUSHL   ARG_VECT                                        ; 0443
                       0000V CF       01  FB 0002D         CALLS   #1, DBG$NOUT_ARG_VECT                           ;
                             50       01  D0 00032         MOVL    #1, R0                                          ; 0445
                                          04 00035         RET                                                     ; 0447

; Routine Size:  54 bytes,    Routine Base:  DBG$CODE + 00E4
;  319           0448  1
```

```
1 GLOBAL ROUTINE DBG$NMAKE_ARG_VECT (ERROR_CODE) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Creates a message argument vector as described on page 4-119 of
1 !       the VAX/VMS system reference, volume 1A.
1 !
1 !       This routine ALWAYS returns the address of a message argument vector.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       error_code        - A longword containing an integer corresponding to a
1 !                           DEBUG message code
1 !
1 !       [fao_count]       - A longword containing the number of fao arguments supplied
1 !                           in conjunction with error_code. This optional parameter
1 !                           MUST be supplied if ANY fao arguments are supplied.
1 !
1 !       [fao_first, ...] - A longword containing an FAO argument to be inserted
1 !                           into the text of a DEBUG message
1 !
1 !               Note that the above sequence may be repeated to construct an argument
1 !               vector for concatenated messages.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword corresponding to the address of a message
1 !       argument vector.
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2    BEGIN
2
2    BUILTIN
2       ACTUALCOUNT,
2       ACTUALPARAMETER;
2
2    LOCAL
2           NUM_ACTUALS,                    ! Number of actual parameters
2           I,                              ! Loop counter
2           ERROR_VECT,                     ! Error vector pointer
```

```
;  378          0506 2               ARG_VECT : REF VECTOR;        ! Messagr argument vector
;  379          0507 2
;  380          0508 2
;  381          0509 2       ! Make the argument vector
;  382          0510 2       !
;  383          0511 2       num_actuals = actualcount ();
;  384          0512 2
;  385          0513 2       arg_vect = dbg$get_tempmem(.num_actuals + 1);
;  386          0514 2       arg_vect [0] = .num_actuals;
;  387          0515 2
;  388          0516 2       INCR i FROM 1 TO .num_actuals
;  389          0517 2       DO
;  390          0518 2           arg_vect [.i] = actualparameter (.i);
;  391          0519 2
;  392          0520 2       RETURN .arg_vect;
;  393          0521 2
;  394          0522 1       END;          ! End of dbg$nmake_arg_vect



                                              0004 00000           .ENTRY  DBG$NMAKE_ARG_VECT, Save R2                              ; 0449
                    52                   6C    9A 00002           MOVZBL  (AP), NUM_ACTUALS                                        ; 0511
                                 01      A2    9F 00005           PUSHAB  1(NUM_ACTUALS)                                           ; 0513
          00000000G 00                   01    FB 00008           CALLS   #1, DBG$GET_TEMPMEM                                      ;
                    60                   52    D0 0000F           MOVL    NUM_ACTUALS, (ARG_VECT)                                  ; 0514
                                         51    D4 00012           CLRL    I                                                        ; 0516
                                         05    11 00014           BRB     2$                                                       ;
                  6041                 6C41    D0 00016 1$:       MOVL    (AP)[I], (ARG_VECT)[I]                                   ; 0518
           F7       51                   52    F3 0001B 2$:       AOBLEQ  NUM_ACTUALS, I, 1$                                       ;
                                               04 0001F           RET                                                              ; 0522

; Routine Size:  32 bytes,    Routine Base:  DBG$CODE + 011A


;  395          0523  1
```

```
1 GLOBAL ROUTINE DBG$NOUT_ARG_VECT (ARGUMENT_VECT) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Outputs the DEBUG error message corresponding to the input message
1 !       argument vector to the user's terminal and/or log file.
1 !
1 !       This routine should be invoked directly only by the DEBUG CLI.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       argument_vect   - A longword containing the address of a message argument
1 !                         vector as described on page 4-119 of the VAX/VMS system
1 !                         reference, volume 1A
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The parameter argument_vect is set to 0 after the output
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Writes a DEBUG error message to the user's terminal and/or log file.
1 !
1 !       This routine signals a debugbug if there is no message to output.
1 !
1 !--
2     BEGIN
2
2
2     ! Check for no error message to output.
2     !
2     IF .argument_vect EQLA 0
2     THEN
2         $DBG_ERROR('DBGNERMSG\DBG$NOUT_ARG_VECT');
2
2     ! Output the message.
2     !
2     SYS$PUTMSG (.argument_vect, dbg$out_message, 0);
2
2     RETURN;
2
1     END;            ! End of dbg$nout_arg_vect
```

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

24  47  42  44  5C  47  53  4D  52  45  4E  47  42  44  1B  00030 P.AAG:  .ASCII  <27>\DBGNERMSG\<92>\DBG$NOUT_ARG_VECT\          ;
        54  43  45  56  5F  47  52  41  5F  54  55  4F  4E  0003F                                                                 ;


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

                                              0000  00000            .ENTRY  DBG$NOUT_ARG_VECT, Save nothing                    ; 0524
                                   04    AC    D5  00002             TSTL    ARGUMENT_VECT                                      ; 0568
                                         15    12  00005             BNEQ    1$                                                 ;
                            00000000'    EF    9F  00007             PUSHAB  P.AAG                                              ; 0570
                                         01    DD  0000D             PUSHL   #1                                                 ;
                            00028362     8F    DD  0000F             PUSHL   #164706                                            ;
                 00000000G  00           03    FB  00015             CALLS   #3, LIB$SIGNAL                                     ;
                                         7E    D4  0001C 1$:         CLRL    -(SP)                                              ; 0574
                            00000000G    00    9F  0001E             PUSHAB  DBG$OUT_MESSAGE                                    ;
                                   04    AC    DD  00024             PUSHL   ARGUMENT_VECT                                      ;
                 00000000G  00           03    FB  00027             CALLS   #3, SYS$PUTMSG                                     ;
                                               04  0002E             RET                                                        ; 0578

; Routine Size:  47 bytes,    Routine Base:  DBG$CODE + 013A


;  452          0579  1
```

```
1 GLOBAL ROUTINE DBG$NSYNTAX_ERROR (WORD_STRING) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Called as a result of the detection of a syntax error. Constructs a
1 !       syntax error message argument vector.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       word_string -           The word corresponding to the syntax error
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The message argument vector associated with the syntax error. This includes
1 !       an ascii string descriiptor which points to the syntax error string.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       The beginning address of the message argument vector
1 !
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2
2     MAP
2       WORD_STRING : REF VECTOR [,BYTE];
2
2     LOCAL
2       ERROR_VECT,                      ! Error message pointer
2       STRING_DESC : REF dbg$stg_desc; ! String descriptor for error message
2
2
2     ! Get storage for the string descriptor
2     !
2     string_desc = dbg$get_tempmem(2);
2
2
2     ! make the string descriptor
2     !
2     string_desc [dsc$w_length] = .word_string [0];
2     string_desc [dsc$a_pointer] = word_string [1];
2
2
2     ! Construct the vector and return it.
2     !
```

```
;  511        0637  2      error_vect = dbg$nmake_arg_vect (dbg$_syntax, 1, .string_desc);
;  512        0638  2
;  513        0639  2      RETURN .error_vect;
;  514        0640  2
;  515        0641  1      END;         ! End of dbg$nsyntax_error



                                        0000 00000        .ENTRY  DBG$NSYNTAX_ERROR, Save nothing              ; 0580
                                      02  DD 00002        PUSHL   #2                                           ; 0626
                     00000000G  00    01  FB 00004        CALLS   #1, DBG$GET_TEMPMEM                          ;
                                60    04  BC  9B 0000B    MOVZBW  @WORD_STRING, (STRING_DESC)                  ; 0631
          04    A0        04    AC    01  C1 0000F        ADDL3   #1, WORD_STRING, 4(STRING_DESC)              ; 0632
                                      50  DD 00015        PUSHL   STRING_DESC                                  ; 0637
                                      01  DD 00017        PUSHL   #1                                           ;
                         00028238     8F  DD 00019        PUSHL   #164408                                      ;
                          8E    AF    03  FB 0001F        CALLS   #3, DBG$NMAKE_ARG_VECT                       ;
                                        04 00023          RET                                                  ; 0641

; Routine Size:  36 bytes,    Routine Base:  DBG$CODE + 0169

;  516        0642  1
;  518          0643  1 END
;  519          0644  0 ELUDOM



                                                         .EXTRN  LIB$SIGNAL

;                                 PSECT SUMMARY
;
;       Name                     Bytes                     Attributes
;
;  DBG$PLIT                         76  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                        397  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          5         0         1000       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1      32          0         0            7       00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1      1545         12         0           97       00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                            418          0         0           31       00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1       386          3         0           22       00:00.3



;                                  COMMAND QUALIFIERS

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

; Size:           397 code + 76 data bytes
; Run Time:          00:13.5
; Elapsed Time:      01:04.6
; Lines/CPU Min:       2872
; Lexemes/CPU-Min:     7378
; Memory Used:   87 pages
; Compilation Complete
```

0087 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY