```
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
```

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  DDDDDDDD    PPPPPPPP      CCCCCCCC
DDDDDDDD    BBBBBBBB      GGGGGGGG  DDDDDDDD    PPPPPPPP      CCCCCCCC
DD      DD  BB      BB  GG          DD      DD  PP      PP  CC
DD      DD  BB      BB  GG          DD      DD  PP      PP  CC
DD      DD  BB      BB  GG          DD      DD  PP      PP  CC
DD      DD  BB      BB  GG          DD      DD  PP      PP  CC
DD      DD  BBBBBBBB    GG          DD      DD  PPPPPPPP    CC
DD      DD  BBBBBBBB    GG          DD      DD  PPPPPPPP    CC
DD      DD  BB      BB  GG  GGGGGG  DD      DD  PP          CC
DD      DD  BB      BB  GG  GGGGGG  DD      DD  PP          CC
DD      DD  BB      BB  GG      GG  DD      DD  PP          CC              ....
DD      DD  BB      BB  GG      GG  DD      DD  PP          CC              ....
DDDDDDDD    BBBBBBBB      GGGGGG    DDDDDDDD    PP            CCCCCCCC      ....
DDDDDDDD    BBBBBBBB      GGGGGG    DDDDDDDD    PP            CCCCCCCC      ....

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 DBGDPC   ( IDENT = 'V04-000') =
;     2          0002  1 BEGIN
;     3          0003  1
;     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
;    29          0029  1 !++
;    30          0030  1 ! FACILITY:     DEBUG (DBG)
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !       Analyzes PC correlation tables for DEBUG.
;    34          0034  1 !
;    35          0035  1 ! ENVIRONMENT:  VAX/VMS, user mode, interrupts disabled.
;    36          0036  1 !
;    37          0037  1 ! AUTHOR:       Carol Peters, CREATION DATE:    16 September 1977
;    38          0038  1 !
;    39          0039  1 ! Version       3.01
;    40          0040  1 !
;    41          0041  1 ! MODIFIED BY:
;    42          0042  1 !       (PS = Ping Sager, RT = Rich Title, JF = John Francis)
;    43          0043  1 !
;    44          0044  1 ! 3.01  15-Sep-81       PS      Correct LINE_END PC address calculation in
;    45          0045  1 !                               PC_TO_LINE_LOOKUP.
;    46          0046  1 ! 3.02  23-Apr-82       RT      Fixed a bug in DBG$PC_TO_LINE_LOOKUP: the routine
;    47          0047  1 !                               was assuming that chasing upscope pointers will
;    48          0048  1 !                               always get you to a routine RST entry.
;    49          0049  1 ! 4.0   13-Dec-82       PS      Switched some old symbolization routines to
;    50          0050  1 !                               use new code.
;    51          0051  1 !        1-Mar-83       JF      Changed return values from DBG$PC_TO_LINE_LOOKUP
;    52          0052  1 !                               so that SUCCESS and FAILURE are shown properly
;    53          0053  1 !       12-Apr-83       RT      Fixed a bug in PC_TO_LINE
;    54          0054  1 !       24-Dec-83       RT      Added comments and did some general cleanup
;    55          0055  1 !--
```

```
;    57          0056  1 ! TABLE OF CONTENTS:
;    58          0057  1 !
;    59          0058  1 FORWARD ROUTINE
;    60          0059  1         dbg$line_to_pc_lookup,  ! Given line number associated it to a PC
;    61          0060  1         dbg$pc_to_line,         ! Matches a PC to a line number
;    62          0061  1         dbg$pc_to_line_lookup,  ! Given PC looks up associated line number
;    63          0062  1         proc_pc_cmd,            ! Processes a string of PC correlation commands
;    64          0063  1         find_eol,               ! Find end of line
;    65          0064  1         give_line_info: NOVALUE;! Give more info about line number
;    66          0065  1
;    67          0066  1
;    68          0067  1 ! INCLUDE FILES:
;    69          0068  1 !
;    70          0069  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    71          0203  1 LIBRARY 'LIB$:DBGGEN.L32';
;    72          0204  1
;    73          0205  1
;    74          0206  1 ! MACROS:
;    75          0207  1 !
;    76          0208  1 MACRO
;    77          0209  1         current_byte    = 0, 0, 8, 1%,          ! current top of record
;    78          0210  1         next_uns_byte   = 1, 0, 8, 0%,          ! byte argument to command
;    79          0211  1         next_uns_word   = 1, 0, 16, 0%,         ! word argument to command
;    80          0212  1         next_uns_long   = 1, 0, 32, 0%,         ! longword argument to command
;    81          0213  1         add_one_byte    = 1, 0, 8, 0%,          ! increment for top of record
;    82          0214  1         add_two_bytes   = 2, 0, 8, 0%,          ! ditto
;    83          0215  1         add_three_bytes = 3, 0, 8, 0%,          ! ditto
;    84          0216  1         add_five_bytes  = 5, 0, 8, 0%;          ! ditto
;    85          0217  1
;    86          0218  1
;    87          0219  1 ! EQUATED SYMBOLS:
;    88          0220  1 !
;    89          0221  1 LITERAL
;    90          0222  1         line_open       = 1,
;    91          0223  1         line_closed     = 2;
;    92          0224  1
;    93          0225  1
;    94          0226  1 ! OWN STORAGE:
;    95          0227  1 !
;    96          0228  1 OWN
;    97          0229  1         dst_entry       : REF dst$record,
;    98          0230  1         dpc_entry       : REF BLOCK [, BYTE],
;    99          0231  1         start_pc,
;   100          0232  1         current_line,
;   101          0233  1         current_stmt,
;   102          0234  1         current_incr,
;   103          0235  1         current_pc,
;   104          0236  1         current_stmt_mode,
;   105          0237  1         current_mark,
;   106          0238  1         prev_line,
;   107          0239  1         prev_stmt,
;   108          0240  1         prev_incr,
;   109          0241  1         prev_pc,
;   110          0242  1         prev_stmt_mode,
;   111          0243  1         prev_mark,
;   112          0244  1         NUM_PC_TBLS,                    ! The number of PC-Correlation DST
;   113          0245  1                                         !       records for the current module
```

```
;  114          0246  1          current_table,
;  115          0247  1          report_next_line,
;  116          0248  1          report_next_stmt,
;  117          0249  1          report_prev_line,
;  118          0250  1          report_prev_stmt,
;  119          0251  1          pctbl_count;
;  120          0252  1
;  121          0253  1
;  122          0254  1 ! EXTERNAL REFERENCES:
;  123          0255  1 !
;  124          0256  1 EXTERNAL ROUTINE
;  125          0257  1          dbg$format_fao_out: NOVALUE,    ! Forward FAO string
;  126          0258  1          dbg$pc_to_symid;                ! Search Moudle SAT to locate RST
;  127          0259  1
```

```
;  129          0260  1 ROUTINE dbg$pc_to_line (match_pc_ptr, modpctbl,    pctbl_base,
;  130          0261  1                          line_no_ptr,  stmt_no_ptr, line_pc) =
;  131          0262  1 !++
;  132          0263  1 ! FUNCTIONAL DESCRIPTION:
;  133          0264  1 !
;  134          0265  1 !       This routine matches an address to a line number.
;  135          0266  1 !       The caller, DBG$PC_TO_LINE_LOOKUP, does the work of finding
;  136          0267  1 !       the PC/LINE table for the module containing the address.
;  137          0268  1 !       A pointer to this table is passed to this routine.
;  138          0269  1 !
;  139          0270  1 !       Each PC correlation record that exists for the module
;  140          0271  1 !       is sequentially analyzed until the desired address
;  141          0272  1 !       is seen.
;  142          0273  1 !
;  143          0274  1 !       See the comments in DBG$PC_TO_LINE_LOOKUP for more details
;  144          0275  1 !       about how this routine is used.
;  145          0276  1 !
;  146          0277  1 ! FORMAL PARAMETERS:
;  147          0278  1 !
;  148          0279  1 !       match_pc_ptr    - The address to be matched.
;  149          0280  1 !       modpctbl        - The address of the table of pointers to
;  150          0281  1 !                         PC/LINE tables in this module. The first
;  151          0282  1 !                         longword of the table is a count of PC/LINE
;  152          0283  1 !                         tables, and the remaining longwords are
;  153          0284  1 !                         pointers to the DST records containing the tables.
;  154          0285  1 !       pctbl_base      - The address which is the base address for
;  155          0286  1 !                         the PC/LINE tables
;  156          0287  1 !       line_no_ptr     - An output parameter for the line number.
;  157          0288  1 !       stmt_no_ptr     - An output parameter for the statement number.
;  158          0289  1 !       line_pc         - An output parameter for the start pc of the
;  159          0290  1 !                         selected line/stmt.
;  160          0291  1 !
;  161          0292  1 ! ROUTINE VALUE:
;  162          0293  1 !
;  163          0294  1 !       This routine returns one of three values: 0, 1, or 2.
;  164          0295  1 !       Note that the caller, DBG$PC_TO_LINE_LOOKUP, may change
;  165          0296  1 !       return status "1" to return status "3" if we did not get
;  166          0297  1 !       an exact match. See that routine for further details
;  167          0298  1 !       on how the return status is used.
;  168          0299  1 !
;  169          0300  1 !       0       - If no match can be made because pc/line tables are
;  170          0301  1 !                 not available for the given address. This may occur
;  171          0302  1 !                 because the module containing the address was not
;  172          0303  1 !                 set or was compiled /NODEBUG, or because the address
;  173          0304  1 !                 is in system space or in an RTL shareable image.
;  174          0305  1 !       1       - If a line number/stmt number was found.
;  175          0306  1 !       2       - If there are pc/line tables available for the
;  176          0307  1 !                 module containing the given address, but no match
;  177          0308  1 !                 was found. This occurs if the address is not within
;  178          0309  1 !                 any line in the module. The use of the "TERM" record
;  179          0310  1 !                 in PC/LINE tables terminates an address range for
;  180          0311  1 !                 a line without starting a new line, and this can
;  181          0312  1 !                 give rise to addresses without line numbers.
;  182          0313  1 !--
;  183          0314  2     BEGIN
;  184          0315  2     MAP
;  185          0316  2         MODPCTBL: REF VECTOR[,LONG];
```

```
;  186          0317  2
;  187          0318  2
;  188          0319  2      ! If we do not have a PC/LINE table, just return 0.
;  189          0320  2      !
;  190          0321  2      IF .MODPCTBL EQL 0 THEN RETURN 0;
;  191          0322  2
;  192          0323  2
;  193          0324  2      ! Set up the OWN variables that we use for reading the PC/LINE tables.
;  194          0325  2      ! This includes a count of the number of PC/LINE DST records in this
;  195          0326  2      ! module we have looked at so far (initialized to 1 here), a count
;  196          0327  2      ! of the total number of PC/LINE DST records in the module, a pointer
;  197          0328  2      ! to our position in the table of PC/LINE DST records,
;  198          0329  2      ! and a pointer to the first such DST record.
;  199          0330  2      ! If there are zero PC/LINE tables in this module, return 0 here.
;  200          0331  2      !
;  201          0332  2      PCTBL_COUNT = 1;
;  202          0333  2      NUM_PC_TBLS = .MODPCTBL[0];
;  203          0334  2      CURRENT_TABLE = MODPCTBL[1];
;  204          0335  2      DST_ENTRY = .MODPCTBL[1];
;  205          0336  2      IF .NUM_PC_TBLS EQL 0 THEN RETURN 0;
;  206          0337  2
;  207          0338  2
;  208          0339  2      ! Initialize the state variables (OWN variables in this module)
;  209          0340  2      ! that are used by PROC_PC_CMD.
;  210          0341  2      !
;  211          0342  2      current_line = 0;
;  212          0343  2      current_stmt = 1;
;  213          0344  2      current_incr = 1;
;  214          0345  2      current_stmt_mode = FALSE;
;  215          0346  2      current_pc = start_pc = .pctbl_base;
;  216          0347  2      current_mark = line_closed;
;  217          0348  2
;  218          0349  2
;  219          0350  2      ! Call a routine that processes all PC correlation commands
;  220          0351  2      ! until a delta-PC command is seen. Then process that
;  221          0352  2      ! delta-PC command and return to this routine. If the processing
;  222          0353  2      ! is generally successful, return 1, otherwise return 0.
;  223          0354  2      !
;  224          0355  2      dpc_entry = dst_entry [dst$b_vflags];
;  225          0356  2      REPEAT
;  226          0357  3          BEGIN
;  227          0358  3          prev_line = .current_line;
;  228          0359  3          prev_stmt = .current_stmt;
;  229          0360  3          prev_incr = .current_incr;
;  230          0361  3          prev_stmt_mode = .current_stmt_mode;
;  231          0362  3          prev_pc = .current_pc;
;  232          0363  3          prev_mark = .current_mark;
;  233          0364  3
;  234          0365  3
;  235          0366  3          ! If we PROC_PC_CMD fails we have come to the end
;  236          0367  3          ! of the PC/LINE table for this module, without finding
;  237          0368  3          ! a match. In this case, return 2, indicating that we
;  238          0369  3          ! are in a module with PC/LINE tables, but we could not
;  239          0370  3          ! match the given PC.
;  240          0371  3          !
;  241          0372  3          IF NOT proc_pc_cmd ( )
;  242          0373  3          THEN
```

```
;  243          0374  3            RETURN 2;
;  244          0375  3
;  245          0376  3
;  246          0377  3        ! Report a match to a line if:
;  247          0378  3        ! - the PC is within the range given by
;  248          0379  3        !   the previous PC and the current PC, and
;  249          0380  3        ! - the line is marked as being OPEN.
;  250          0381  3        !
;  251          0382  3        IF (.prev_pc LEQA .match_pc_ptr) AND
;  252          0383  3           (.match_pc_ptr LSSA .current_pc) AND
;  253          0384  4           (.prev_mark EQL line_open)
;  254          0385  3        THEN
;  255          0386  4            BEGIN
;  256          0387  5            .stmt_no_ptr = (IF .prev_stmt EQL 1 THEN 0
;  257          0388  4                                               ELSE .prev_stmt); ! Huh?
;  258          0389  4            .line_no_ptr = .prev_line;
;  259          0390  4            .line_pc = .prev_pc;
;  260          0391  4            RETURN 1;
;  261          0392  3            END;
;  262          0393  3
;  263          0394  3
;  264          0395  3        ! Found nothing this round; continue trying.
;  265          0396  3        !
;  266          0397  2        END;            ! End of REPEAT.
;  267          0398  2
;  268          0399  2
;  269          0400  2    ! We have not found a match - return 2, indicating that we
;  270          0401  2    ! are in a module with PC/LINE tables, but we could not
;  271          0402  2    ! match the given PC.
;  272          0403  2    !
;  273          0404  2    RETURN 2;
;  274          0405  1    END;


                                        .TITLE  DBGDPC
                                        .IDENT  \V04-000\

                                        .PSECT  DBG$OWN,NOEXE,  PIC,2

                                00000 DST_ENTRY:
                                        .BLKB   4
                                00004 DPC_ENTRY:
                                        .BLKB   4
                                00008 START_PC:
                                        .BLKB   4
                                0000C CURRENT_LINE:
                                        .BLKB   4
                                00010 CURRENT_STMT:
                                        .BLKB   4
                                00014 CURRENT_INCR:
                                        .BLKB   4
                                00018 CURRENT_PC:
                                        .BLKB   4
                                0001C CURRENT_STMT_MODE:
                                        .BLKB   4
                                00020 CURRENT_MARK:
                                        .BLKB   4
```

```
                                   00024 PREV_LINE:
                                              .BLKB    4
                                   00028 PREV_STMT:
                                              .BLKB    4
                                   0002C PREV_INCR:
                                              .BLKB    4
                                   00030 PREV_PC:.BLKB    4
                                   00034 PREV_STMT_MODE:
                                              .BLKB    4
                                   00038 PREV_MARK:
                                              .BLKB    4
                                   0003C NUM_PC_TBLS:
                                              .BLKB    4
                                   00040 CURRENT_TABLE:
                                              .BLKB    4
                                   00044 REPORT_NEXT_LINE:
                                              .BLKB    4
                                   00048 REPORT_NEXT_STMT:
                                              .BLKB    4
                                   0004C REPORT_PREV_LINE:
                                              .BLKB    4
                                   00050 REPORT_PREV_STMT:
                                              .BLKB    4
                                   00054 PCTBL_COUNT:
                                              .BLKB    4

                                              .EXTRN   DBG$FORMAT_FAO_OUT
                                              .EXTRN   DBG$PC_TO_SYMID

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

                              0004 00000 DBG$PC_TO_LINE:
                                              .WORD    Save R2                                    ; 0260
           52 00000000'  E    9E 00002        MOVAB    NUM_PC_TBLS, R2                            ;
           50       08   AC   D0 00009        MOVL     MODPCTBL, R0                               ; 0321
                         13   13 0000D        BEQL     1$                                         ;
      18   A2            01   D0 0000F        MOVL     #1, PCTBL_COUNT                            ; 0332
           62            60   D0 00013        MOVL     (R0), NUM_PC_TBLS                          ; 0333
      04   A2       04   A0   9E 00016        MOVAB    4(R0), CURRENT_TABLE                       ; 0334
      C4   A2       04   A0   D0 0001B        MOVL     4(R0), DST_ENTRY                           ; 0335
                         62   D5 00020        TSTL     NUM_PC_TBLS                                ; 0336
                         73   13 00022 1$:    BEQL     6$                                         ;
                    D0   A2   D4 00024        CLRL     CURRENT_LINE                               ; 0342
      D4   A2            01   D0 00027        MOVL     #1, CURRENT_STMT                           ; 0343
      D8   A2            01   D0 0002B        MOVL     #1, CURRENT_INCR                           ; 0344
                    E0   A2   D4 0002F        CLRL     CURRENT_STMT_MODE                          ; 0345
           50       0C   AC   D0 00032        MOVL     PCTBL_BASE, R0                             ; 0346
      CC   A2            50   D0 00036        MOVL     R0, START_PC                               ;
      DC   A2            50   D0 0003A        MOVL     R0, CURRENT_PC                             ;
      E4   A2            02   D0 0003E        MOVL     #2, CURRENT_MARK                           ; 0347
C8 A2 C4   A2            02   C1 00042        ADDL3    #2, DST_ENTRY, DPC_ENTRY                   ; 0355
      E8   A2       D0   A2   7D 00048 2$:    MOVQ     CURRENT_LINE, PREV_LINE                    ; 0358
      F8   A2       E0   A2   7D 0004D        MOVQ     CURRENT_STMT_MODE, PREV_STMT_MODE          ; 0361
      F0   A2       D8   A2   7D 00052        MOVQ     CURRENT_INCR, PREV_INCR                    ; 0360
   0000V   CF            00   FB 00057        CALLS    #0, PROC_PC_CMD                            ; 0372
           34            50   E9 0005C        BLBC     R0, 5$                                     ;
      04   AC       F4   A2   D1 0005F        CMPL     PREV_PC, MATCH_PC_PTR                      ; 0382
```

```
                              E2  1A 00064          BGTRU   2$                                         ;
        DC    A2       04     AC  D1 00066          CMPL    MATCH_PC_PTR, CURRENT_PC                   ; 0383
                              DB  1E 0006B          BGEQU   2$                                         ;
              01       FC     A2  D1 0006D          CMPL    PREV_MARK, #1                              ; 0384
                              D5  12 00071          BNEQ    2$                                         ;
              01       EC     A2  D1 00073          CMPL    PREV_STMT, #1                              ; 0387
                              04  12 00077          BNEQ    3$                                         ;
                              50  D4 00079          CLRL    R0                                         ;
                              04  11 0007B          BRB     4$                                         ;
              50       EC     A2  D0 0007D 3$:      MOVL    PREV_STMT, R0                              ; 0388
        14    BC              50  D0 00081 4$:      MOVL    R0, @STMT_NO_PTR                           ; 0387
        10    BC       E8     A2  D0 00085          MOVL    PREV_LINE, @LINE_NO_PTR                    ; 0389
        18    BC       F4     A2  D0 0008A          MOVL    PREV_PC, @LINE_PC                          ; 0390
              50              01  D0 0008F          MOVL    #1, R0                                     ; 0391
                                  04 00092          RET                                                ;
              50              02  D0 00093 5$:      MOVL    #2, R0                                     ; 0404
                                  04 00096          RET                                                ;
                              50  D4 00097 6$:      CLRL    R0                                         ; 0405
                                  04 00099          RET                                                ;

; Routine Size:  154 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  276     0406  1 GLOBAL ROUTINE DBG$LINE_TO_PC_LOOKUP (LINE_NUM, STMT_NUM, MC_PTR,
;  277     0407  1                                        LINE_PC, LINE_END, FLAG) =
;  278     0408  1 !
;  279     0409  1 ! FUNCTIONAL DESCRIPTION:
;  280     0410  1 !       This routine finds the absolute PC address associated with
;  281     0411  1 !       a line number/statement number.
;  282     0412  1 !
;  283     0413  1 !       Each PC correlation record that exists for a single routine
;  284     0414  1 !       is sequentially analyzed until the desired line number
;  285     0415  1 !       is seen.
;  286     0416  1 !
;  287     0417  1 !       If a match cannot be made because an end of routine record or
;  288     0418  1 !       an invalid record is recognized, then this routine returns
;  289     0419  1 !       FALSE.
;  290     0420  1 !
;  291     0421  1 ! FORMAL PARAMETERS:
;  292     0422  1 !       line_num        - the line number to find.
;  293     0423  1 !       stmt_num        - the statement number to find.
;  294     0424  1 !       mc_ptr          - module rstptr
;  295     0425  1 !       line_pc         - where to store the computed address.
;  296     0426  1 !       line_end        - a copy-back pointer for the line-end pc value.
;  297     0427  1 !       flag            - flag set to indicate more line information is needed.
;  298     0428  1 !
;  299     0429  1 ! ROUTINE VALUE:
;  300     0430  1 !       The routine value is TRUE if the desired line was successfully
;  301     0431  1 !               found; it is FALSE otherwise.
;  302     0432  1 !
;  303     0433  1 !
;  304     0434  1
;  305     0435  2     BEGIN
;  306     0436  2     MAP
;  307     0437  2         MC_PTR: REF RST$ENTRY;
;  308     0438  2
;  309     0439  2     LOCAL
;  310     0440  2         MODPCTBL: REF VECTOR[,LONG];
;  311     0441  2
;  312     0442  2
;  313     0443  2     ! Adjust a statement number of 1 to 0 (%LINE 10.1 is equivalent
;  314     0444  2     ! to %LINE 10, and the algorithm below coughs at statement numbers of 1
;  315     0445  2     !
;  316     0446  2     IF .STMT_NUM EQL 1 THEN STMT_NUM = 0;
;  317     0447  2
;  318     0448  2
;  319     0449  2     ! Set up the OWN variables that we use for reading the PC/LINE tables.
;  320     0450  2     ! This includes a count of the number of PC/LINE DST records in this
;  321     0451  2     ! module we have looked at so far (initialized to 1 here), a count
;  322     0452  2     ! of the total number of PC/LINE DST records in the module, a pointer
;  323     0453  2     ! to our position in the table of PC/LINE DST records,
;  324     0454  2     ! and a pointer to the first such DST record.
;  325     0455  2     ! If there are zero PC/LINE tables in this module, return 0 here.
;  326     0456  2     !
;  327     0457  2     PCTBL_COUNT = 1;
;  328     0458  2     MODPCTBL = .MC_PTR[RST$L_MODPCTBL];
;  329     0459  2     IF .MODPCTBL EQL 0 THEN RETURN FALSE;
;  330     0460  2     NUM_PC_TBLS = .MODPCTBL[0];
;  331     0461  2     CURRENT_TABLE = MODPCTBL[1];
;  332     0462  2     DST_ENTRY = .MODPCTBL[1];
```

```
;  333          0463  2      IF .NUM_PC_TBLS EQL 0 THEN RETURN 0;
;  334          0464  2
;  335          0465  2
;  336          0466  2      ! Initialize state variables. These are OWN variables that
;  337          0467  2      ! are used by PROC_PC_CMD.
;  338          0468  2      !
;  339          0469  2      current_line = 0;
;  340          0470  2      current_stmt = 1;
;  341          0471  2      current_incr = 1;
;  342          0472  2      current_stmt_mode = FALSE;
;  343          0473  2      current_pc = start_pc = .mc_ptr[rst$l_pctbl_base];
;  344          0474  2      current_mark = line_closed;
;  345          0475  2
;  346          0476  2
;  347          0477  2      ! Loop through the PC Correlation Tables for this module until the
;  348          0478  2      ! desired line number is found or the table ends.  To do this, we call
;  349          0479  2      ! PROC_PC_CMD to process all PC Correlation commands until a delta-PC
;  350          0480  2      ! command is found.  It then returns a PC and a line number and we
;  351          0481  2      ! check whether that is the line number we are looking for.  If not,
;  352          0482  2      ! we loop for the next line until the desired line is found or no PC
;  353          0483  2      ! Correlation commands remain.
;  354          0484  2      !
;  355          0485  2      dpc_entry = dst_entry [dst$b_vflags];
;  356          0486  2      REPORT_PREV_LINE = 0;
;  357          0487  2      REPORT_PREV_STMT = 1;
;  358          0488  2      REPORT_NEXT_LINE = .LINE_NUM;
;  359          0489  2      REPORT_NEXT_STMT = .STMT_NUM;
;  360          0490  2      WHILE TRUE DO
;  361          0491  3          BEGIN
;  362          0492  3
;  363          0493  3
;  364          0494  3          ! Remember the previous values of all the state variables
;  365          0495  3          ! before getting the current values this time around.
;  366          0496  3          !
;  367          0497  3          PREV_LINE = .CURRENT_LINE;
;  368          0498  3          PREV_STMT = .CURRENT_STMT;
;  369          0499  3          PREV_INCR = .CURRENT_INCR;
;  370          0500  3          PREV_STMT_MODE = .CURRENT_STMT_MODE;
;  371          0501  3          PREV_PC = .CURRENT_PC;
;  372          0502  3          PREV_MARK = .CURRENT_MARK;
;  373          0503  3
;  374          0504  3
;  375          0505  3          ! Call PROC_PC_CMD to get the next PC - line number pair.
;  376          0506  3          ! When there are no more lines, exit this loop.
;  377          0507  3          !
;  378          0508  3          IF NOT PROC_PC_CMD() THEN EXITLOOP;
;  379          0509  3
;  380          0510  3
;  381          0511  3          ! Set report next line and stmt for the first time.
;  382          0512  3          !
;  383          0513  3          IF (.REPORT_NEXT_LINE EQL .LINE_NUM) AND
;  384          0514  4             (.REPORT_NEXT_STMT EQL .STMT_NUM)
;  385          0515  3          THEN
;  386          0516  4              BEGIN
;  387          0517  4              IF (.CURRENT_LINE GTR .LINE_NUM) OR
;  388          0518  5                 ((.CURRENT_LINE EQL .LINE_NUM) AND
;  389          0519  5                  (.CURRENT_STMT GTR .STMT_NUM))
```

```
;  390     0520  4            THEN
;  391     0521  5                BEGIN
;  392     0522  5                REPORT_NEXT_LINE = .CURRENT_LINE;
;  393     0523  5                REPORT_NEXT_STMT = .CURRENT_STMT;
;  394     0524  4                END;
;  395     0525  4
;  396     0526  3            END;
;  397     0527  3
;  398     0528  3
;  399     0529  3        ! At this point we have Prev. line, current line, and given line info.
;  400     0530  3        ! So we define the reporting line information centered around given line.
;  401     0531  3        ! (we choose the closest two ends value).
;  402     0532  3        !
;  403     0533  3        ! Define report prev. line.
;  404     0534  3        !
;  405     0535  3        IF .REPORT_PREV_LINE LSS .LINE_NUM
;  406     0536  3        THEN
;  407     0537  4            BEGIN
;  408     0538  4            IF .PREV_LINE LSS .LINE_NUM
;  409     0539  4            THEN
;  410     0540  4                REPORT_PREV_LINE = MAX(.REPORT_PREV_LINE, .PREV_LINE)
;  411     0541  4            ELSE
;  412     0542  5                BEGIN
;  413     0543  6                IF ((.PREV_LINE EQL .LINE_NUM) AND
;  414     0544  6                    (.PREV_STMT LSS .STMT_NUM))
;  415     0545  5                THEN
;  416     0546  6                    BEGIN
;  417     0547  6                    REPORT_PREV_LINE = .PREV_LINE;
;  418     0548  6                    REPORT_PREV_STMT = .PREV_STMT;
;  419     0549  5                    END;
;  420     0550  5
;  421     0551  4                END;
;  422     0552  4
;  423     0553  4            END
;  424     0554  4
;  425     0555  3        ELSE
;  426     0556  4            BEGIN
;  427     0557  5            IF ((.REPORT_PREV_LINE EQL .LINE_NUM) AND
;  428     0558  5                (.REPORT_PREV_STMT LSS .STMT_NUM))
;  429     0559  4            THEN
;  430     0560  5                BEGIN
;  431     0561  5                IF (.PREV_LINE EQL .LINE_NUM) AND
;  432     0562  6                   (.PREV_STMT LSS .STMT_NUM)
;  433     0563  5                THEN
;  434     0564  5                    REPORT_PREV_STMT = MAX(.PREV_STMT, .REPORT_PREV_STMT);
;  435     0565  5
;  436     0566  4                END;
;  437     0567  4
;  438     0568  3            END;
;  439     0569  3
;  440     0570  3
;  441     0571  3        ! Define report next line.
;  442     0572  3        !
;  443     0573  3        IF .REPORT_NEXT_LINE GTR .LINE_NUM
;  444     0574  3        THEN
;  445     0575  4            BEGIN
;  446     0576  4            IF .CURRENT_LINE GTR .LINE_NUM
```

```
;  447     0577  4            THEN
;  448     0578  4                REPORT_NEXT_LINE = MIN(.REPORT_NEXT_LINE, .CURRENT_LINE)
;  449     0579  4            ELSE
;  450     0580  5                BEGIN
;  451     0581  6                IF ((.CURRENT_LINE EQL .LINE_NUM) AND
;  452     0582  6                    (.CURRENT_STMT GTR .STMT_NUM))
;  453     0583  5                THEN
;  454     0584  6                    BEGIN
;  455     0585  6                    REPORT_NEXT_LINE = .CURRENT_LINE;
;  456     0586  6                    REPORT_NEXT_STMT = .CURRENT_STMT;
;  457     0587  5                    END;
;  458     0588  5
;  459     0589  4                END;
;  460     0590  4
;  461     0591  4            END
;  462     0592  4
;  463     0593  3        ELSE
;  464     0594  4            BEGIN
;  465     0595  5            IF ((.REPORT_NEXT_LINE EQL .LINE_NUM) AND
;  466     0596  5                (.REPORT_NEXT_STMT GTR .STMT_NUM))
;  467     0597  4            THEN
;  468     0598  5                BEGIN
;  469     0599  5                IF (.CURRENT_LINE EQL .LINE_NUM) AND
;  470     0600  6                   (.CURRENT_STMT GTR .STMT_NUM)
;  471     0601  5                THEN
;  472     0602  5                    REPORT_PREV_STMT = MIN(.CURRENT_STMT, .REPORT_NEXT_STMT);
;  473     0603  5
;  474     0604  4                END;
;  475     0605  4
;  476     0606  3            END;
;  477     0607  3
;  478     0608  3
;  479     0609  3        ! Note that: above code did not take care of the equality condition.
;  480     0610  3        ! it should be set up here and tested in give_line_info.
;  481     0611  3        !
;  482     0612  3        ! If the current line number is equivalent to the one we were
;  483     0613  3        ! passed (and this includes the statement number), then we
;  484     0614  3        ! return the corresponding PC to LINE_PC and we return TRUE.
;  485     0615  3        ! If we are at the right line but there is no such statement
;  486     0616  3        ! number, we clear LINE_PC and return FALSE.
;  487     0617  3        !
;  488     0618  3        IF .CURRENT_LINE EQL .LINE_NUM
;  489     0619  3        THEN
;  490     0620  4            BEGIN
;  491     0621  4            IF  MAX (.CURRENT_STMT, 1) EQL MAX(.STMT_NUM, 1)
;  492     0622  4            THEN
;  493     0623  5                BEGIN
;  494     0624  5                .LINE_PC = .CURRENT_PC;
;  495     0625  5                IF NOT FIND_EOL(.LINE_END)
;  496     0626  6                THEN
;  497     0627  6                    BEGIN
;  498     0628  6                    IF .FLAG THEN GIVE_LINE_INFO(.LINE_NUM, .STMT_NUM);
;  499     0629  6                    RETURN FALSE;
;  500     0630  5                    END;
;  501     0631  5
;  502     0632  5                RETURN TRUE;
;  503     0633  5                END
```

```
                                  5
                          ELSE    4
                              BEGIN   5
                              IF MAX(.CURRENT_STMT,1) GTR MAX(.STMT_NUM,1)   5
                              THEN    5
                                  BEGIN   6
                                  .LINE_PC = 0;   6
                                  IF .FLAG THEN GIVE_LINE_INFO(.LINE_NUM, .STMT_NUM);   6
                                  RETURN FALSE;   6
                                  END;    5
                                          5
                              END;    4
                                          4
                          END;    3
                                          3
                      END;                            ! End of WHILE loop over PC Corr Tbl   2
                                          2
                                          2
                  ! The desired line number was not found.  Clear LINE_PC and return FALSE   2
                  ! as the routine value.   2
                  !   2
                  IF .FLAG THEN GIVE_LINE_INFO(.LINE_NUM, .STMT_NUM);   2
                  .LINE_PC = 0;   2
                  RETURN FALSE;   2
                  END;    1
```

```
                                   001C 00000        .ENTRY  DBG$LINE_TO_PC_LOOKUP, Save R2,R3,R4          ; 0406
               54 00000000'    EF    9E 00002        MOVAB   CURRENT_STMT, R4                              ;
               01          08  AC    D1 00009        CMPL    STMT_NUM, #1                                  ; 0446
                               03    12 0000D        BNEQ    1$                                            ;
                           08  AC    D4 0000F        CLRL    STMT_NUM                                      ;
          44   A4              01    D0 00012 1$:    MOVL    #1, PCTBL_COUNT                               ; 0457
               50          0C  AC    D0 00016        MOVL    MC_PTR, R0                                    ; 0458
               51          2C  A0    D0 0001A        MOVL    44(R0), MODPCTBL                              ;
                               11    13 0001E        BEQL    2$                                            ; 0459
          2C   A4              61    D0 00020        MOVL    (MODPCTBL), NUM_PC_TBLS                       ; 0460
          30   A4          04  A1    9E 00024        MOVAB   4(MODPCTBL), CURRENT_TABLE                    ; 0461
          F0   A4          04  A1    D0 00029        MOVL    4(MODPCTBL), DST_ENTRY                        ; 0462
                           2C  A4    D5 0002E        TSTL    NUM_PC_TBLS                                   ; 0463
                               03    12 00031 2$:    BNEQ    3$                                            ;
                             0170    31 00033        BRW     26$                                           ;
                           FC  A4    D4 00036 3$:    CLRL    CURRENT_LINE                                  ; 0469
               64              01    D0 00039        MOVL    #1, CURRENT_STMT                              ; 0470
          04   A4              01    D0 0003C        MOVL    #1, CURRENT_INCR                              ; 0471
                           0C  A4    D4 00040        CLRL    CURRENT_STMT_MODE                             ; 0472
               50          1C  A0    D0 00043        MOVL    28(R0), R0                                    ; 0473
          F8   A4              50    D0 00047        MOVL    R0, START_PC                                  ;
          08   A4              50    D0 0004B        MOVL    R0, CURRENT_PC                                ;
          10   A4              02    D0 0004F        MOVL    #2, CURRENT_MARK                              ; 0474
  F4   A4 F0   A4              02    C1 00053        ADDL3   #2, DST_ENTRY, DPC_ENTRY                      ; 0485
                           3C  A4    D4 00059        CLRL    REPORT_PREV_LINE                              ; 0486
          40   A4              01    D0 0005C        MOVL    #1, REPORT_PREV_STMT                          ; 0487
               52          04  AC    7D 00060        MOVQ    LINE_NUM, R2                                  ; 0488
```

```
                34   A4          52   7D 00064          MOVQ    R2, REPORT_NEXT_LINE
                14   A4    FC    A4   7D 00068 4$:      MOVQ    CURRENT_LINE, PREV_LINE                 ; 0497
                24   A4    0C    A4   7D 0006D          MOVQ    CURRENT_STMT_MODE, PREV_STMT_MODE       ; 0500
                1C   A4    04    A4   7D 00072          MOVQ    CURRENT_INCR, PREV_INCR                 ; 0499
          0000V CF               00   FB 00077          CALLS   #0, PROC_PC_CMD                         ; 0508
                03               50   E8 0007C          BLBS    R0, 5$
                              0116   31 0007F          BRW     24$
                52         34    A4   D1 00082 5$:      CMPL    REPORT_NEXT_LINE, R2                    ; 0513
                                 1E   12 00086          BNEQ    7$
                53         38    A4   D1 00088          CMPL    REPORT_NEXT_STMT, R3                    ; 0514
                                 18   12 0008C          BNEQ    7$
                50         FC    A4   D0 0008E          MOVL    CURRENT_LINE, R0                        ; 0517
                52               50   D1 00092          CMPL    R0, R2
                                 07   14 00095          BGTR    6$
                                 0D   12 00097          BNEQ    7$                                      ; 0518
                53               64   D1 00099          CMPL    CURRENT_STMT, R3                        ; 0519
                                 08   15 0009C          BLEQ    7$
                34   A4          50   D0 0009E 6$:      MOVL    R0, REPORT_NEXT_LINE                    ; 0522
                38   A4          64   D0 000A2          MOVL    CURRENT_STMT, REPORT_NEXT_STMT          ; 0523
                50         3C    A4   D0 000A6 7$:      MOVL    REPORT_PREV_LINE, R0                    ; 0535
                52               50   D1 000AA          CMPL    R0, R2
                                 2A   18 000AD          BGEQ    10$
                51         14    A4   D0 000AF          MOVL    PREV_LINE, R1                           ; 0538
                52               51   D1 000B3          CMPL    R1, R2
                                 0E   18 000B6          BGEQ    9$
                51               50   D1 000BB          CMPL    R0, R1                                  ; 0540
                                 03   18 000BB          BGEQ    8$
                50               51   D0 000BD          MOVL    R1, R0
          3C    A4               50   D0 000C0 8$:      MOVL    R0, REPORT_PREV_LINE
                                 39   11 000C4          BRB     12$
                                 37   12 000C6 9$:      BNEQ    12$                                     ; 0543
                53         18    A4   D1 000C8          CMPL    PREV_STMT, R3                           ; 0544
                                 31   18 000CC          BGEQ    12$
          3C    A4               51   D0 000CE          MOVL    R1, REPORT_PREV_LINE                    ; 0547
          40    A4         18    A4   D0 000D2          MOVL    PREV_STMT, REPORT_PREV_STMT             ; 0548
                                 26   11 000D7          BRB     12$                                     ; 0535
                                 24   12 000D9 10$:     BNEQ    12$                                     ; 0557
                53         40    A4   D1 000DB          CMPL    REPORT_PREV_STMT, R3                    ; 0558
                                 1E   18 000DF          BGEQ    12$
                52         14    A4   D1 000E1          CMPL    PREV_LINE, R2                           ; 0561
                                 18   12 000E5          BNEQ    12$
                53         18    A4   D1 000E7          CMPL    PREV_STMT, R3                           ; 0562
                                 12   18 000EB          BGEQ    12$
                50         18    A4   D0 000ED          MOVL    PREV_STMT, R0                           ; 0564
          40    A4               50   D1 000F1          CMPL    R0, REPORT_PREV_STMT
                                 04   18 000F5          BGEQ    11$
                50         40    A4   D0 000F7          MOVL    REPORT_PREV_STMT, R0
          40    A4               50   D0 000FB 11$:     MOVL    R0, REPORT_PREV_STMT
                50         34    A4   D0 000FF 12$:     MOVL    REPORT_NEXT_LINE, R0                    ; 0573
                52               50   D1 00103          CMPL    R0, R2
                                 28   15 00106          BLEQ    15$
                51         FC    A4   D0 00108          MOVL    CURRENT_LINE, R1                        ; 0576
                52               51   D1 0010C          CMPL    R1, R2
                                 0E   15 0010F          BLEQ    14$
                51               50   D1 00111          CMPL    R0, R1                                  ; 0578
                                 03   15 00114          BLEQ    13$
                50               51   D0 00116          MOVL    R1, R0
```

```
      34   A4            50  D0 00119 13$:    MOVL   RC, REPORT_NEXT_LINE               ;
                         35  11 0011D         BRB    1'$                                ;
                         33  12 0011F 14$:    BNEQ   17$                                ; 0581
           53            64  D1 00121         CMPL   CURRENT_STMT, R3                   ; 0582
                         2E  15 00124         BLEQ   17$                                ;
      34   A4            51  D0 00126         MOVL   R1, REPORT_NEXT_LINE               ; 0585
      38   A4            64  D0 0012A         MOVL   CURRENT_STMT, REPORT_NEXT_STMT     ; 0586
                         24  11 0012E         BRB    17$                                ; 0573
                         22  12 00130 15$:    BNEQ   17$                                ; 0595
           53      38    A4  D1 00132         CMPL   REPORT_NEXT_STMT, R3               ; 0596
                         1C  15 00136         BLEQ   17$                                ;
           52      FC    A4  D1 00138         CMPL   CURRENT_LINE, R2                   ; 0599
                         16  12 0013C         BNEQ   17$                                ;
           53            64  D1 0013E         CMPL   CURRENT_STMT, R3                   ; 0600
                         11  15 00141         BLEQ   17$                                ;
           50            64  D0 00143         MOVL   CURRENT_STMT, R0                   ; 0602
      38   A4            50  D1 00146         CMPL   R0, REPORT_NEXT_STMT               ;
                         04  15 0014A         BLEQ   16$                                ;
           50      38    A4  D0 0014C         MOVL   REPORT_NEXT_STMT, R0               ;
      40   A4            50  D0 00150 16$:    MOVL   R0, REPORT_PREV_STMT               ;
           52      FC    A4  D1 00154 17$:    CMPL   CURRENT_LINE, R2                   ; 0618
                         03  13 00158         BEQL   19$                                ;
                       FF0B  31 0015A 18$:    BRW    4$                                 ;
           51            64  D0 0015D 19$:    MOVL   CURRENT_STMT, R1                   ; 0621
                         03  14 00160         BGTR   20$                                ;
           51            01  D0 00162         MOVL   #1, R1                             ;
           50            53  D0 00165 20$:    MOVL   R3, R0                             ;
                         03  14 00168         BGTR   21$                                ;
           50            01  D0 0016A         MOVL   #1, R0                             ;
           50            51  D1 0016D 21$:    CMPL   R1, R0                             ;
                         14  12 00170         BNEQ   22$                                ;
      10   BC      08    A4  D0 00172         MOVL   CURRENT_PC, @LINE_PC               ; 0624
                   14    AC  DD 00177         PUSHL  LINE_END                           ; 0625
   0000V   CF            01  FB 0017A         CALLS  #1, FIND_EOL                       ;
           09            50  E, 0017F         BLBC   R0, 23$                            ;
           50            01  D0 00182         MOVL   #1, R0                             ; 0632
                             04 00185         RET                                       ;
                         D2  15 00186 22$:    BLEQ   18$                                ; 0637
                   10    BC  D4 00188         CLRL   @LINE_PC                           ; 0640
           17      18    AC  E9 0018B 23$:    BLBC   FLAG, 26$                          ; 0641
                         0C  BB 0018F         PUSHR  #^M<R2,R3>                         ;
   0000V   CF            02  FB 00191         CALLS  #2, GIVE_LINE_INFO                 ;
                         0E  11 00196         BRB    26$                                ; 0642
           07      18    AC  E9 00198 24$:    BLBC   FLAG, 25$                          ; 0655
                         0C  BB 0019C         PUSHR  #^M<R2,R3>                         ;
   0000V   CF            02  FB 0019E         CALLS  #2, GIVE_LINE_INFO                 ;
                   10    BC  D4 001A3 25$:    CLRL   @LINE_PC                           ; 0656
                         50  D4 001A6 26$:    CLRL   R0                                 ; 0658
                             04 001A8         RET                                       ;

; Routine Size:  425 bytes,    Routine Base:  DBG$CODE + 009A
```

```
;  530          0659  1 GLOBAL ROUTINE dbg$pc_to_line_lookup (match_pc_ptr, line_no_ptr, stmt_no_ptr,
;  531          0660  1                                       line_start,  line_end,    mod_symid) =
;  532          0661  1 !
;  533          0662  1 ! FUNCTIONAL DESCRIPTION:
;  534          0663  1 !
;  535          0664  1 !       This routine matches an address to a line number.
;  536          0665  1 !       We need to do this in several situations:
;  537          0666  1 !
;  538          0667  1 !       1. When stepping by line, to determine when to stop stepping. (DBGEVENT)
;  539          0668  1 !       2. When symbolizing a code address to put out "%LINE XX" (DBGSYMBLZ)
;  540          0669  1 !       3. Putting out the SHOW CALLS display (DBGTBK)
;  541          0670  1 !       4. Finding the start of the line for "EX/INS ^" (DBGLEVEL3)
;  542          0671  1 !       5. Source display, as in EX/SOURCE .PC (DBGSOURCE)
;  543          0672  1 !
;  544          0673  1 !       The line number (and statement number, for BASIC) is returned.
;  545          0674  1 !       Also returned are: the start and end address of the line,
;  546          0675  1 !       and a pointer to the module RST entry for the module containing
;  547          0676  1 !       the given address.
;  548          0677  1 !
;  549          0678  1 !       Each PC correlation record that exists for the module
;  550          0679  1 !       is sequentially analyzed until the desired address is seen.
;  551          0680  1 !
;  552          0681  1 !       This routine is actually just a cover routine for DBG$PC_TO_LINE,
;  553          0682  1 !       where the real work is done.
;  554          0683  1 !
;  555          0684  1 ! FORMAL PARAMETERS:
;  556          0685  1 !
;  557          0686  1 !       match_pc_ptr    - the address to be matched.
;  558          0687  1 !       line_no_ptr     - an output parameter for the line number.
;  559          0688  1 !       stmt_no_ptr     - an output parameter for the statement number.
;  560          0689  1 !       line_start      - an output parameter for the start pc of the
;  561          0690  1 !                         selected line/stmt.
;  562          0691  1 !       line_end        - an output parameter for the end pc of the
;  563          0692  1 !                         selected line/stmt.
;  564          0693  1 !       mod_symid       - An in/out parameter, as follows:
;  565          0694  1 !
;  566          0695  1 !                         If the caller has a SYMID for a block, routine,
;  567          0696  1 !                         or module which contains the given address, then
;  568          0697  1 !                         this symid can be passed in here. This saves
;  569          0698  1 !                         a search of the Static Address Table.
;  570          0699  1 !                         If the caller
;  571          0700  1 !                         does not have a symid, then zero is passed in.
;  572          0701  1 !                         Note that these are passed in with an extra level
;  573          0702  1 !                         of indirection, e.g.,
;  574          0703  1 !                         SYMID = 0;
;  575          0704  1 !                         STATUS = DBG$PC_TO_LINE_LOOKUP(.ADDRESS,...,SYMID);
;  576          0705  1 !
;  577          0706  1 !                         In either case, this parameter is filled in with
;  578          0707  1 !                         the address of the module containing MATCH_PC_PTR.
;  579          0708  1 !
;  580          0709  1 ! ROUTINE VALUE:
;  581          0710  1 !
;  582          0711  1 !       This routine can return four values: 0, 1, 2, or 3.
;  583          0712  1 !       Most of the callers just test the result for
;  584          0713  1 !       TRUE (meaning a match was found), or FALSE (meaning a match
;  585          0714  1 !       was not found). So for these callers, 0 and 2 are the same,
;  586          0715  1 !       and 1 and 3 are the same.
```

```
;  587          0716  1 !
;  588          0717  1 !       DBGEVENT needs more detailed information than just whether
;  589          0718  1 !       a match was found, in order to decide whether to continue
;  590          0719  1 !       stepping. It needs to know why a match was not found, or
;  591          0720  1 !       if one was found, whether or not it was an exact match.
;  592          0721  1 !       So for the DBGEVENT call, we return the following:
;  593          0722  1 !
;  594          0723  1 !       0       - If no match can be made because pc/line tables are
;  595          0724  1 !                 not available for the given address. This may occur
;  596          0725  1 !                 because the module containing the address was not
;  597          0726  1 !                 set or was compiled /NODEBUG, or because the address
;  598          0727  1 !                 is in system space or in an RTL shareable image.
;  599          0728  1 !       1       - If a line number/stmt number was found, and we
;  600          0729  1 !                 have an exact match to that line number.
;  601          0730  1 !       2       - If there are pc/line tables available for the
;  602          0731  1 !                 module containing the given address, but no match
;  603          0732  1 !                 was found. This occurs if the address is not within
;  604          0733  1 !                 any line in the module. The use of the "TERM" record
;  605          0734  1 !                 in PC/LINE tables terminates an address range for
;  606          0735  1 !                 a line without starting a new line, and this can
;  607          0736  1 !                 give rise to addresses without line numbers.
;  608          0737  1 !       3       - If there is a line number associated with the address, but
;  609          0738  1 !                 it is not an exact match.
;  610          0739  1 !
;  611          0740  2     BEGIN
;  612          0741  2     LOCAL
;  613          0742  2         rstptr: REF rst$entry,  : Module RST pointer
;  614          0743  2         status;                 ! Return Status
;  615          0744  2
;  616          0745  2
;  617          0746  2     ! If we do not know an RST entry for a program unit
;  618          0747  2     ! containing the given address, we'll look
;  619          0748  2     ! it up through the Program-level SAT.
;  620          0749  2     ! If we already have the information
;  621          0750  2     ! (passed in from the caller) then just set it up.
;  622          0751  2     !
;  623          0752  2     IF ..mod_symid EQL 0
;  624          0753  2     THEN
;  625          0754  3         BEGIN
;  626          0755  3         status = dbg$pc_to_symid(.match_pc_ptr, rstptr);
;  627          0756  3
;  628          0757  3
;  629          0758  3         ! If PC_TO_SYMID failed, then we do not have a module containing
;  630          0759  3         ! the address in our module chain. Thus, return zero.
;  631          0760  3         !
;  632          0761  3         IF NOT .status THEN RETURN 0;
;  633          0762  3         END
;  634          0763  3
;  635          0764  2     ELSE
;  636          0765  2         rstptr = ..mod_symid;
;  637          0766  2
;  638          0767  2
;  639          0768  2     ! Go upscope to the module level, just in case a caller passed in
;  640          0769  2     ! a routine or block RST entry.
;  641          0770  2     !
;  642          0771  2     WHILE (.rstptr[rst$b_kind] NEQ rst$k_module) DO
;  643          0772  2         rstptr = .rstptr[rst$l_upscopeptr];
```

```
2
2
2      ! Set the return module RST.
2      !
2      .mod_symid = .rstptr;
2
2
2      ! Now call the routine to do the real work. Pass along the three
2      ! output parameters LINE_NO_PTR, STMT_NO_PTR, and LINE_START,
2      ! to be filled in by DBG$PC_TO_LINE.
2      !
2      status = dbg$pc_to_line(.match_pc_ptr, .rstptr[rst$l_modpctbl],
2                              .rstptr[rst$l_pctbl_base],
2                              .line_no_ptr,  .stmt_no_ptr, .line_start);
2
2
2      ! We get the return code from DBG$PC_TO_LINE. Here we check
2      ! for the PC being an exact match. If not, we change the "1"
2      ! return status to a "3" to indicate this. We also fill in the
2      ! LINE_END output parameter, using the OWN variable CURRENT_PC
2      ! that gets set in the processing of PC/LINE records.
2      !
2      IF .status EQL 1
2      THEN
3          BEGIN
3          .line_end = .current_pc - 1;
3          IF ..line_start NEQA .match_pc_ptr
3          THEN
3              status = 3;          ! not exact match.
2          END;
2
2      RETURN .status;
1      END;
```

```
                              0000 00000       .ENTRY  DBG$PC_TO_LINE_LOOKUP, Save nothing        ; 0659
           5E              04   C2 00002       SUBL2   #4, SP                                     ;
                     18    BC   D5 00005       TSTL    @MOD_SYMID                                 ; 0752
                           11   12 00008       BNEQ    1$                                         ;
                           5E   DD 0000A       PUSHL   SP                                         ; 0755
                     04    AC   DD 0000C       PUSHL   MATCH_PC_PTR                               ;
 00000000G 00              02   FB 0000F       CALLS   #2, DBG$PC_TO_SYMID                        ;
           06              50   E8 00016       BLBS    STATUS, 2$                                 ; 0761
                           48   11 00019       BRB     4$                                         ;
           6E        18    BC   D0 0001B 1$:   MOVL    @MOD_SYMID, RSTPTR                         ; 0765
           51              6E   D0 0001F 2$:   MOVL    RSTPTR, R1                                 ; 0771
           01        14    A1   91 00022       CMPB    20(R1), #1                                 ;
                           06   13 00026       BEQL    3$                                         ;
           6E        10    A1   D0 00028       MOVL    16(R1), RSTPTR                             ; 0772
                           F1   11 0002C       BRB     2$                                         ;
           51              6E   D0 0002E 3$:   MOVL    RSTPTR, R1                                 ; 0777
        18 BC              51   D0 00031       MOVL    R1, @MOD_SYMID                             ;
           7E        0C    AC   7D 00035       MOVQ    STMT_NO_PTR, -(SP)                         ; 0786
                     08    AC   DD 00039       PUSHL   LINE_NO_PTR                                ;
```

```
                                   1C   A1  DD 0003C        PUSHL   28(R1)                                  ; 0785
                                   2C   A1  DD 0003F        PUSHL   44(R1)                                  ; 0784
                                   04   AC  DD 00042        PUSHL   MATCH_PC_PTR                            ;
                   FD73    CF           06  FB 00045        CALLS   #6, DBG$PC_TO_LINE                      ;
                           01           50  D1 0004A        CMPL    STATUS, #1                              ; 0795
                                        16  12 0004D        BNEQ    5$                                      ;
         14   BC 00000000'  EF          01  C3 0004F        SUBL3   #1, CURRENT_PC, @LINE_END               ; 0798
                     04     AC     10   BC  D1 00058        CMPL    @LINE_START, MATCH_PC_PTR               ; 0799
                                        06  13 0005D        BEQL    5$                                      ;
                           50           03  D0 0005F        MOVL    #3, STATUS                              ; 0801
                                            04 00062        RET                                             ; 0804
                                        50  D4 00063 4$:    CLRL    R0                                      ; 0805
                                            04 00065 5$:    RET                                             ;

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 0243
```

```
;  678     0806  1 ROUTINE PROC_PC_CMD =
;  679     0807  1 !++
;  680     0808  1 ! Functional description:
;  681     0809  1 !
;  682     0810  1 !       This routine processes PC correlation commands until a
;  683     0811  1 !       delta-Pc command is seen. The delta-PC command is also processed.
;  684     0812  1 !       Then this routine returns with all the contents of the
;  685     0813  1 !       parameter pointers updated appropriately.
;  686     0814  1 !
;  687     0815  1 !       This routine moves from PC record to PC record as necessary. If
;  688     0816  1 !       no more records are seen, this routine returns false. If
;  689     0817  1 !       an error is seen in a PC correlation record, then this
;  690     0818  1 !       routine sets the contents of line_ptr to zero and
;  691     0819  1 !       returns false.
;  692     0820  1 !
;  693     0821  1 ! Inputs:
;  694     0822  1 !
;  695     0823  1 ! Implicit inputs:
;  696     0824  1 !       None
;  697     0825  1 !
;  698     0826  1 ! Implicit outputs:
;  699     0827  1 !       the contents of the line pointer, the increment pointer, the
;  700     0828  1 !       statement pointer, the next_pc pointer, dpc_entry, and possible
;  701     0829  1 !       dst_entry are updated to new values.
;  702     0830  1 !
;  703     0831  1 ! Routine value:
;  704     0832  1 !       TRUE if all goes well, otherwise FALSE.
;  705     0833  1 !
;  706     0834  1 ! Side effects:
;  707     0835  1 !       More of the correlation records for this routine are read.
;  708     0836  1 !--
;  709     0837  1
;  710     0838  2         BEGIN
;  711     0839  2
;  712     0840  2         REPEAT
;  713     0841  3                 BEGIN
;  714     0842  3
;  715     0843  3
;  716     0844  3                 ! See whether the current record is exhausted. If
;  717     0845  3                 ! so, get a new record. If none are available,
;  718     0846  3                 ! return FALSE. Otherwise, set dpc_entry to point to
;  719     0847  3                 ! the address of the third byte of the correlation record.
;  720     0848  3                 !
;  721     0849  4                 IF dpc_entry[current_byte] GTR (.dst_entry[dst$b_length] +
;  722     0850  4                                 dst_entry[dst$b_length])
;  723     0851  3                 THEN
;  724     0852  4                         BEGIN
;  725     0853  4                         PCTBL_COUNT = .PCTBL_COUNT + 1;
;  726     0854  4                         IF .PCTBL_COUNT GTR .NUM_PC_TBLS THEN RETURN FALSE;
;  727     0855  4                         current_table = .current_table + 4;
;  728     0856  4                         dst_entry = ..current_table;
;  729     0857  4                         dpc_entry = dst_entry [dst$b_vflags];
;  730     0858  3                         END;
;  731     0859  3
;  732     0860  3
;  733     0861  3                 ! Now process each command, either PC correlation or
;  734     0862  3                 ! delta-PC one at a time. Once a delta-PC command is
```

```
;  735     0863  3                ! processed, control returns from this routine to its
;  736     0864  3                ! caller.
;  737     0865  3                !
;  738     0866  3                CASE .dpc_entry [current_byte] FROM 1 TO dst$k_pccor_high OF
;  739     0867  3                        SET
;  740     0868  3
;  741     0869  3                        ! Read the next two bytes as an unsigned word
;  742     0870  3                        ! representing a delta-PC value.  Update the next_pc
;  743     0871  3                        ! and update the dpc_entry address.
;  744     0872  3                        !
;  745     0873  3                        [dst$k_delta_pc_w]:
;  746     0874  3                                BEGIN
;  747     0875  4                                IF .current_stmt_mode
;  748     0876  4                                THEN
;  749     0877  4                                        current_stmt = .current_stmt + 1
;  750     0878  4                                ELSE
;  751     0879  4                                        current_line = .current_line +
;  752     0880  4                                                        .current_incr;
;  753     0881  4
;  754     0882  4                                current_mark = line_open;
;  755     0883  4                                current_pc = .current_pc +
;  756     0884  4                                                .dpc_entry [next_uns_word];
;  757     0885  4                                dpc_entry = dpc_entry [add_three_bytes];
;  758     0886  4                                RETURN TRUE;
;  759     0887  4                                END;
;  760     0888  3
;  761     0889  3
;  762     0890  3                        ! Read the next four bytes as an unsigned longword
;  763     0891  3                        ! representing a delta-PC value.  Update the next_pc
;  764     0892  3                        ! and update the dpc_entry address.
;  765     0893  3                        !
;  766     0894  3                        [dst$k_delta_pc_l]:
;  767     0895  3                                BEGIN
;  768     0896  4                                IF .current_stmt_mode
;  769     0897  4                                THEN
;  770     0898  4                                        current_stmt = .current_stmt + 1
;  771     0899  4                                ELSE
;  772     0900  4                                        current_line = .current_line +
;  773     0901  4                                                        .current_incr;
;  774     0902  4
;  775     0903  4                                current_mark = line_open;
;  776     0904  4                                current_pc = .current_pc +
;  777     0905  4                                                .dpc_entry [next_uns_long];
;  778     0906  4                                dpc_entry = dpc_entry [add_five_bytes];
;  779     0907  4                                RETURN TRUE;
;  780     0908  4                                END;
;  781     0909  3
;  782     0910  3
;  783     0911  3                        ! Increase the current line number by the value
;  784     0912  3                        ! contained in the next unsigned byte.
;  785     0913  3                        !
;  786     0914  3                        [dst$k_incr_linum]:
;  787     0915  3                                BEGIN
;  788     0916  4                                current_line = .current_line + .dpc_entry [next_uns_byte];
;  789     0917  4                                IF .current_stmt_mode THEN current_stmt = 1;
;  790     0918  4                                dpc_entry = dpc_entry [add_two_bytes];
;  791     0919  4
```

```
;  792          0920  3                END;
;  793          0921  3
;  794          0922  3
;  795          0923  3            ! Increase the current line number by the value
;  796          0924  3            ! contained in the next unsigned word.
;  797          0925  3            !
;  798          0926  3            [dst$k_incr_linum_w]:
;  799          0927  4                    BEGIN
;  800          0928  4                    IF .current_stmt_mode THEN current_stmt = 1;
;  801          0929  4                    current_line = .current_line + .dpc_entry [next_uns_word];
;  802          0930  4                    dpc_entry = dpc_entry [add_three_bytes];
;  803          0931  3                    END;
;  804          0932  3
;  805          0933  3
;  806          0934  3            ! Increase the current line number by the value
;  807          0935  3            ! contained in the next unsigned longword.
;  808          0936  3            !
;  809          0937  3            [dst$k_incr_linum_l]:
;  810          0938  4                    BEGIN
;  811          0939  4                    IF .current_stmt_mode THEN current_stmt = 1;
;  812          0940  4                    current_line = .current_line + .dpc_entry [next_uns_long];
;  813          0941  4                    dpc_entry = dpc_entry [add_five_bytes];
;  814          0942  3                    END;
;  815          0943  3
;  816          0944  3
;  817          0945  3            ! Change the line increment from its present value to
;  818          0946  3            ! the value contained in the next unsigned byte.
;  819          0947  3            !
;  820          0948  3            [dst$k_set_linum_incr]:
;  821          0949  4                    BEGIN
;  822          0950  4                    IF .current_stmt_mode THEN current_stmt = 1;
;  823          0951  4                    current_incr = .dpc_entry [next_uns_byte];
;  824          0952  4                    dpc_entry = dpc_entry [add_two_bytes];
;  825          0953  3                    END;
;  826          0954  3
;  827          0955  3
;  828          0956  3            ! Change the line increment from its present value to
;  829          0957  3            ! the value contained in the next word.
;  830          0958  3            !
;  831          0959  3            [dst$k_set_linum_incr_w]:
;  832          0960  4                    BEGIN
;  833          0961  4                    IF .current_stmt_mode THEN current_stmt = 1;
;  834          0962  4                    current_incr = .dpc_entry [next_uns_word];
;  835          0963  4                    dpc_entry = dpc_entry [add_three_bytes];
;  836          0964  3                    END;
;  837          0965  3
;  838          0966  3
;  839          0967  3            ! Revert to a line increment of value 1.
;  840          0968  3            !
;  841          0969  3            [dst$k_reset_linum_incr]:
;  842          0970  4                    BEGIN
;  843          0971  4                    IF .current_stmt_mode THEN current_stmt = 1;
;  844          0972  4                    current_incr = 1;
;  845          0973  4                    dpc_entry = dpc_entry [add_one_byte];
;  846          0974  3                    END;
;  847          0975  3
;  848          0976  3            [dst$k_beg_stmt_mode]:
```

```
;  849          0977  4                        BEGIN
;  850          0978  4                        IF .current_mark NEQ line_open
;  851          0979  4                        THEN
;  852          0980  4                                SIGNAL(dbg$_invdstrec);
;  853          0981  4
;  854          0982  4                        current_stmt = 1;
;  855          0983  4                        current_stmt_mode = TRUE;
;  856          0984  4                        dpc_entry = dpc_entry[add_one_byte];
;  857          0985  3                        END;
;  858          0986  3
;  859          0987  3                [dst$k_end_stmt_mode]:
;  860          0988  4                        BEGIN
;  861          0989  4                        current_stmt = 1;
;  862          0990  4                        current_stmt_mode = FALSE;
;  863          0991  4                        dpc_entry = dpc_entry[add_one_byte];
;  864          0992  3                        END;
;  865          0993  3
;  866          0994  3                [dst$k_set_linum_b]:
;  867          0995  4                        BEGIN
;  868          0996  4                        current_line = .dpc_entry[next_uns_byte];
;  869          0997  4                        dpc_entry = dpc_entry[add_two_bytes];
;  870          0998  3                        END;
;  871          0999  3
;  872          1000  3                [dst$k_set_linum]:
;  873          1001  4                        BEGIN
;  874          1002  4                        current_line = .dpc_entry[next_uns_word];
;  875          1003  4                        dpc_entry = dpc_entry[add_three_bytes];
;  876          1004  3                        END;
;  877          1005  3
;  878          1006  3                [dst$k_set_linum_l]:
;  879          1007  4                        BEGIN
;  880          1008  4                        current_line = .dpc_entry[next_uns_long];
;  881          1009  4                        dpc_entry = dpc_entry[add_five_bytes];
;  882          1010  3                        END;
;  883          1011  3
;  884          1012  3                [dst$k_set_stmtnum]:
;  885          1013  4                        BEGIN
;  886          1014  4                        current_stmt = .dpc_entry[next_uns_word];
;  887          1015  4                        dpc_entry = dpc_entry[add_three_bytes];
;  888          1016  3                        END;
;  889          1017  3
;  890          1018  3                [dst$k_set_pc]:
;  891          1019  4                        BEGIN
;  892          1020  4                        IF .current_mark NEQ line_closed
;  893          1021  4                        THEN
;  894          1022  4                                SIGNAL (dbg$_invdstrec);
;  895          1023  4
;  896          1024  4                        current_pc = .start_pc +
;  897          1025  4                                        .dpc_entry[next_uns_byte];
;  898          1026  4                        dpc_entry = dpc_entry[add_two_bytes];
;  899          1027  3                        END;
;  900          1028  3
;  901          1029  3                [dst$k_set_pc_w]:
;  902          1030  4                        BEGIN
;  903          1031  4                        IF .current_mark NEQ line_closed
;  904          1032  4                        THEN
;  905          1033  4                                SIGNAL (dbg$_invdstrec);
```

```
;  906          1034  4                        current_pc = .start_pc +
;  907          1035  4                                     .dpc_entry[next_uns_word];
;  908          1036  4                        dpc_entry = dpc_entry[add_three_bytes];
;  909          1037  4                        END;
;  910          1038  3
;  911          1039  3
;  912          1040  3                [dst$k_set_pc_l]:
;  913          1041  4                        BEGIN
;  914          1042  4                        IF .current_mark NEQ line_closed
;  915          1043  4                        THEN
;  916          1044  4                                SIGNAL (dbg$_invdstrec);
;  917          1045  4
;  918          1046  4                        current_pc = .start_pc +
;  919          1047  4                                     .dpc_entry[next_uns_long];
;  920          1048  4                        dpc_entry = dpc_entry[add_five_bytes];
;  921          1049  4                        END;
;  922          1050  4
;  923          1051  4
;  924          1052  4                ! Set the current PC value to an absolute address.
;  925          1053  4                !
;  926          1054  3                [DST$K_SET_ABS_PC]:
;  927          1055  4                        BEGIN
;  928          1056  4                        IF .CURRENT_MARK NEQ LINE_CLOSED
;  929          1057  4                        THEN
;  930          1058  4                            SIGNAL(DBG$_INVDSTREC);
;  931          1059  4
;  932          1060  4                        CURRENT_PC = .DPC_ENTRY[NEXT_UNS_LONG];
;  933          1061  4                        DPC_ENTRY = DPC_ENTRY[ADD_FIVE_BYTES];
;  934          1062  3                        END;
;  935          1063  3
;  936          1064  3                [dst$k_term]:
;  937          1065  4                        BEGIN
;  938          1066  4                        current_pc = .current_pc +
;  939          1067  4                                     .dpc_entry[next_uns_byte];
;  940          1068  4                        current_mark = line_closed;
;  941          1069  4                        dpc_entry = dpc_entry[add_two_bytes];
;  942          1070  4                        RETURN TRUE;
;  943          1071  3                        END;
;  944          1072  3
;  945          1073  3                [dst$k_term_w]:
;  946          1074  4                        BEGIN
;  947          1075  4                        current_pc = .current_pc +
;  948          1076  4                                     .dpc_entry[next_uns_word];
;  949          1077  4                        current_mark = line_closed;
;  950          1078  4                        dpc_entry = dpc_entry[add_three_bytes];
;  951          1079  4                        RETURN TRUE;
;  952          1080  3                        END;
;  953          1081  3
;  954          1082  3
;  955          1083  3                [dst$k_term_l]:
;  956          1084  4                        BEGIN
;  957          1085  4                        current_pc = .current_pc +
;  958          1086  4                                     .dpc_entry[next_uns_long];
;  959          1087  4                        current_mark = line_closed;
;  960          1088  4                        dpc_entry = dpc_entry[add_five_bytes];
;  961          1089  4                        RETURN TRUE;
;  962          1090  3                        END;
```

```
3
3
3                       ! This is a standard delta_PC command if the value is
3                       ! less than or equal to zero. Otherwise it is an error.
3                       ! If okay, set next_pc value, update the dpc_entry,
3                       ! and return with success.
3                       !
3                       [OUTRANGE]:
4                               BEGIN
4                               IF .dpc_entry [current_byte] LSS
4                                                       dst$k_delta_pc_low
4                               OR .dpc_entry[current_byte] GTR
4                                                       dst$k_delta_pc_high
4                               THEN
4                                       SIGNAL (dbg$_invdstrec);
4
4                               IF .current_stmt_mode
4                               THEN
4                                       current_stmt = .current_stmt + 1
4                               ELSE
4                                       current_line = .current_line +
4                                                               .current_incr;
4
4                               current_pc = .current_pc -
4                                               .dpc_entry [current_byte];
4                               current_mark = line_open;
4                               dpc_entry = dpc_entry [add_one_byte];
4                               RETURN TRUE;
3                               END;
3
3                       TES;
2               END;
2
2       RETURN 0;
1       END;
```

```
                                  001C 0000u PROC_PC_CMD:
                                                  .WORD   Save R2,R3,R4                          ; 0806
          54 00000000G   00   9E 00002           MOVAB   LIB$SIGNAL, R4                         ;
          53 00000000'   EF   9E 00009           MOVAB   DPC_ENTRY, R3                          ;
          50          FC B3   9A 00010 1$:       MOVZBL  @DST_ENTRY, R0                         ; 0850
          50          FC A3   C0 00014           ADDL2   DST_ENTRY, R0                          ;
          50             63   D1 00018           CMPL    DPC_ENTRY, R0                          ; 0849
                      1B      15 0001B           BLEQ    3$                                     ;
                   50    A3   D6 0001D           INCL    PCTBL_COUNT                            ; 0853
     38   A3       50    A3   D1 00020           CMPL    PCTBL_COUNT, NUM_PC_TBLS               ; 0854
                      03      15 00025           BLEQ    2$                                     ;
                    01ED      31 00027           BRW     56$                                    ;
     3C   A3          04      C0 0002A 2$:       ADDL2   #4, CURRENT_TABLE                      ; 0855
     FC   A3       3C    B3   D0 0002E           MOVL    @CURRENT_TABLE, DST_ENTRY              ; 0856
63   FC   A3          02      C1 00033           ADDL3   #2, DST_ENTRY, DPC_ENTRY               ; 0857
          52             63   D0 00038 3$:       MOVL    DPC_ENTRY, R2                          ; 0866
14        01             62   8F 0003B           CASEB   (R2), #1, #20                          ;
```

```
                00C2    00A1    008F    0055    0003F 4$:       .WORD   8$-4$,-                                          ;
                0107    00EE    00E0    00D1    00047                   16$-4$,-                                        ;
                0170    014F    012E    0119    0004F                   17$-4$,-                                        ;
                018B    01B9    01A8    0127    00057                   21$-4$,-                                        ;
                0120    0112    00B3    0075    0005F                   23$-4$,-                                        ;
                                        01C8    00067                   25$-4$,-                                        ;
                                                                        27$-4$,-                                        ;
                                                                        29$-4$,-                                        ;
                                                                        33$-4$,-                                        ;
                                                                        38$-4$,-                                        ;
                                                                        41$-4$,-                                        ;
                                                                        45$-4$,-                                        ;
                                                                        36$-4$,-                                        ;
                                                                        51$-4$,-                                        ;
                                                                        52$-4$,-                                        ;
                                                                        47$-4$,-                                        ;
                                                                        13$-4$,-                                        ;
                                                                        19$-4$,-                                        ;
                                                                        31$-4$,-                                        ;
                                                                        34$-4$,-                                        ;
                                                                        53$-4$                                          ;
                                                62  95  00069           TSTB    (R2)                                    ; 1102
                                                09  15  0006B           BLEQ    5$                                      ;
                                    0002832A    8F  DD  0006D           PUSHL   #164650                                 ; 1105
                                64              01  FB  00073           CALLS   #1, LIB$SIGNAL                          ;
                                05      18      A3  E9  00076 5$:       BLBC    CURRENT_STMT_MODE, 6$                   ; 1107
                                        0C      A3  D6  0007A           INCL    CURRENT_STMT                            ; 1109
                                                05  11  0007D           BRB     7$                                      ;
                        08      A3      10      A3  C0  0007F 6$:       ADDL2   CURRENT_INCR, CURRENT_LINE              ; 1112
                                50      00      B3  98  00084 7$:       CVTBL   @DPC_ENTRY, R0                          ; 1115
                        14      A3              50  C2  00088           SUBL2   R0, CURRENT_PC                          ;
                        1C      A3              01  D0  0008C           MOVL    #1, CURRENT_MARK                        ; 1116
                                                63  D6  00090           INCL    DPC_ENTRY                               ; 1117
                                                1D  11  00092           BRB     12$                                     ; 1118
                                05      18      A3  E9  00094 8$:       BLBC    CURRENT_STMT_MODE, 9$                   ; 0876
                                        0C      A3  D6  00098           INCL    CURRENT_STMT                            ; 0878
                                                05  11  0009B           BRB     10$                                     ;
                        08      A3      10      A3  C0  0009D 9$:       ADDL2   CURRENT_INCR, CURRENT_LINE              ; 0881
                        1C      A3              01  D0  000A2 10$:      MOVL    #1, CURRENT_MARK                        ; 0883
                                50      01      A2  3C  000A6           MOVZWL  1(R2), R0                               ; 0885
                        14      A3              50  C0  000AA           ADDL2   R0, CURRENT_PC                          ;
                                63              03  C0  000AE 11$:      ADDL2   #3, DPC_ENTRY                           ; 0886
                                        015F    31  00B1 12$:           BRW     55$                                     ; 0887
                                05      18      A3  E9  000B4 13$:      BLBC    CURRENT_STMT_MODE, 14$                  ; 0897
                                        0C      A3  D6  000B8           INCL    CURRENT_STMT                            ; 0899
                                                05  11  000BB           BRB     15$                                     ;
                        08      A3      10      A3  C0  000BD 14$:      ADDL2   CURRENT_INCR, CURRENT_LINE              ; 0902
                        1C      A3              01  D0  000C2 15$:      MOVL    #1, CURRENT_MARK                        ; 0904
                        14      A3      01      A2  C0  000C6           ADDL2   1(R2), CURRENT_PC                       ; 0906
                                        0142    31  00CB           BRW     54$                                     ; 0907
                                50      01      A2  9A  000CE 16$:      MOVZBL  1(R2), R0                               ; 0917
                        08      A3              50  C0  000D2           ADDL2   R0, CURRENT_LINE                        ;
                                7C      18      A3  E9  000D6           BLBC    CURRENT_STMT_MODE, 32$                  ; 0918
                        0C      A3              01  D0  000DA           MOVL    #1, CURRENT_STMT                        ;
                                                76  11  000DE           BRB     32$                                     ; 0919
                                04      18      A3  E9  000E0 17$:      BLBC    CURRENT_STMT_MODE, 18$                  ; 0928
                        0C      A3              01  D0  000E4           MOVL    #1, CURRENT_STMT                        ;
```

```
                50        01    A2  3C 000E8 18$:    MOVZWL  1(R2), R0                       ; 0929
          08    A3              50  C0 000EC          ADDL2   R0, CURRENT_LINE                ;
                                79  11 000F0          BRB     37$                             ; 0930
                04        18    A3  E9 000F2 19$:    BLBC    CURRENT_STMT_MODE, 20$          ; 0939
          0C    A3              01  D0 000F6          MOVL    #1, CURRENT_STMT                ;
          08    A3        01    A2  C0 000FA 20$:    ADDL2   1(R2), CURRENT_LINE             ; 0940
                                63  11 000FF          BRB     35$                             ; 0941
                04        18    A3  E9 00101 21$:    BLBC    CURRENT_STMT_MODE, 22$          ; 0950
          0C    A3              01  D0 00105          MOVL    #1, CURRENT_STMT                ;
          10    A3        01    A2  9A 00109 22$:    MOVZBL  1(R2), CURRENT_INCR             ; 0951
                                79  11 0010E          BRB     40$                             ; 0952
                04        18    A3  E9 00110 23$:    BLBC    CURRENT_STMT_MODE, 24$          ; 0961
          0C    A3              01  D0 00114          MOVL    #1, CURRENT_STMT                ;
          10    A3        01    A2  3C 00118 24$:    MOVZWL  1(R2), CURRENT_INCR             ; 0962
                                4C  11 0011D          BRB     37$                             ; 0963
                04        18    A3  E9 0011F 25$:    BLBC    CURRENT_STMT_MODE, 26$          ; 0971
          0C    A3              01  D0 00123          MOVL    #1, CURRENT_STMT                ;
          10    A3              01  D0 00127 26$:    MOVL    #1, CURRENT_INCR                ; 0972
                                20  11 0012B          BRB     30$                             ; 0973
                01        1C    A3  D1 0012D 27$:    CMPL    CURRENT_MARK, #1                ; 0978
                                09  13 00131          BEQL    28$                             ;
                    0002832A    8F  DD 00133          PUSHL   #164650                         ; 0980
                64              01  FB 00139          CALLS   #1, LIB$SIGNAL                  ;
          0C    A3              01  D0 0013C 28$:    MOVL    #1, CURRENT_STMT                ; 0982
          18    A3              01  D0 00140          MOVL    #1, CURRENT_STMT_MODE           ; 0983
                                07  11 00144          BRB     30$                             ; 0984
          0C    A3              01  D0 00146 29$:    MOVL    #1, CURRENT_STMT                ; 0989
                      18        A3  D4 0014A          CLRL    CURRENT_STMT_MODE               ; 0990
                                63  D6 0014D 30$:    INCL    DPC_ENTRY                       ; 0991
                                5C  11 0014F          BRB     44$                             ; 0866
          08    A3        01    A2  9A 00151 31$:    MOVZBL  1(R2), CURRENT_LINE             ; 0996
                                31  11 00156 32$:    BRB     40$                             ; 0997
          08    A3        01    A2  3C 00158 33$:    MOVZWL  1(R2), CURRENT_LINE             ; 1002
                                4B  11 0015D          BRB     43$                             ; 1003
          08    A3        01    A2  D0 0015F 34$:    MOVL    1(R2), CURRENT_LINE             ; 1008
                                7B  11 00164 35$:    BRB     49$                             ; 1009
          0C    A3        01    A2  3C 00166 36$:    MOVZWL  1(R2), CURRENT_STMT             ; 1014
                                3D  11 0016B 37$:    BRB     43$                             ; 1015
                02        1C    A3  D1 0016D 38$:    CMPL    CURRENT_MARK, #2                ; 1020
                                09  13 00171          BEQL    39$                             ;
                    0002832A    8F  DD 00173          PUSHL   #164650                         ; 1022
                64              01  FB 00179          CALLS   #1, LIB$SIGNAL                  ;
                50              63  D0 0017C 39$:    MOVL    DPC_ENTRY, R0                   ; 1025
                51        01    A0  9A 0017F          MOVZBL  1(R0), R1                       ;
          14    A3        04 B341   9E 00183          MOVAB   @START_PC[R1], CURRENT_PC       ;
                63              02  C0 00189 40$:    ADDL2   #2, DPC_ENTRY                   ; 1026
                                56  11 0018C          BRB     50$                             ; 0866
                02        1C    A3  D1 0018E 41$:    CMPL    CURRENT_MARK, #2                ; 1031
                                09  13 00192          BEQL    42$                             ;
                    0002832A    8F  DD 00194          PUSHL   #164650                         ; 1033
                64              01  FB 0019A          CALLS   #1, LIB$SIGNAL                  ;
                50              63  D0 0019D 42$:    MOVL    DPC_ENTRY, R0                   ; 1036
                51        01    A0  3C 001A0          MOVZWL  1(R0), R1                       ;
          14    A3        04 B341   9E 001A4          MOVAB   @START_PC[R1], CURRENT_PC       ;
                63              03  C0 001AA 43$:    ADDL2   #3, DPC_ENTRY                   ; 1037
                                35  11 001AD 44$:    BRB     50$                             ; 0866
                02        1C    A3  D1 001AF 45$:    CMPL    CURRENT_MARK, #2                ; 1042
```

```
                                 09  13 001B3        BEQL    46$                                   ;
                        0002832A 8F  DD 001B5        PUSHL   #164650                               ; 1044
                    64           01  FB 001BB        CALLS   #1, LIB$SIGNAL                        ;
                    50           63  D0 001BE 46$:   MOVL    DPC_ENTRY, R0                         ; 1047
    14   A3     04  A3      01   A0  C1 001C1        ADDL3   1(R0), START_PC, CURRENT_PC           ;
                                 17  11 001C8        BRB     49$                                   ; 1048
                    02      1C   A3  D1 001CA 47$:   CMPL    CURRENT_MARK, #2                      ; 1056
                                 09  13 001CE        BEQL    48$                                   ;
                        0002832A 8F  DD 001D0        PUSHL   #164650                               ; 1058
                    64           01  FB 001D6        CALLS   #1, LIB$SIGNAL                        ;
                    50           63  D0 001D9 48$:   MOVL    DPC_ENTRY, R0                         ; 1060
                14  A3      01   A0  D0 001DC        MOVL    1(R0), CURRENT_PC                     ;
                    63           05  C0 001E1 49$:   ADDL2   #5, DPC_ENTRY                         ; 1061
                               FE29  31 001E4 50$:   BRW     1$                                    ; 0866
                    50      01   A2  9A 001E7 51$:   MOVZBL  1(R2), R0                             ; 1067
                14  A3           50  C0 001EB        ADDL2   R0, CURRENT_PC                        ;
                1C  A3           02  D0 001EF        MOVL    #2, CURRENT_MARK                      ; 1068
                    63           02  C0 001F3        ADDL2   #2, DPC_ENTRY                         ; 1069
                                 1B  11 001F6        BRB     55$                                   ; 1070
                    50      01   A2  3C 001F8 52$:   MOVZWL  1(R2), R0                             ; 1076
                14  A3           50  C0 001FC        ADDL2   R0, CURRENT_PC                        ;
                1C  A3           02  D0 00200        MOVL    #2, CURRENT_MARK                      ; 1077
                               FEA7  31 00204        BRW     11$                                   ; 1078
                14  A3      01   A2  C0 00207 53$:   ADDL2   1(R2), CURRENT_PC                     ; 1086
                1C  A3           02  D0 0020C        MOVL    #2, CURRENT_MARK                      ; 1087
                    63           05  C0 00210 54$:   ADDL2   #5, DPC_ENTRY                         ; 1088
                    50           01  D0 00213 55$:   MOVL    #1, R0                                ; 1089
                                     04 00216        RET                                           ;
                                 50  D4 00217 56$:   CLRL    R0                                    ; 1125
                                     04 00219        RET                                           ;

; Routine Size:  538 bytes,    Routine Base:  DBG$CODE + 02A9
```

```
;  999     1126  1 ROUTINE FIND_EOL(LINE_END) =
; 1000     1127  1 !++
; 1001     1128  1 ! Functional description:
; 1002     1129  1 !       This routine processes PC correlation commands until
; 1003     1130  1 !       an end of line is found.
; 1004     1131  1 !
; 1005     1132  1 ! Inputs:
; 1006     1133  1 !       line_end        - a copy-back pointer for the value of the end-of-line
; 1007     1134  1 !
; 1008     1135  1 ! Implicit inputs:
; 1009     1136  1 !       None
; 1010     1137  1 !
; 1011     1138  1 ! Implicit outputs:
; 1012     1139  1 !       the contents of the line pointer, the increment pointer, the
; 1013     1140  1 !       statement pointer, the next_pc pointer, dpc_entry, and possible
; 1014     1141  1 !       dst_entry are updated to new values.
; 1015     1142  1 !
; 1016     1143  1 ! Routine value:
; 1017     1144  1 !       TRUE if all goes well, otherwise FALSE.
; 1018     1145  1 !
; 1019     1146  1 ! Side effects:
; 1020     1147  1 !       More of the correlation records for this routine are read.
; 1021     1148  1 !--
; 1022     1149  1
; 1023     1150  2         BEGIN
; 1024     1151  2
; 1025     1152  2         REPEAT
; 1026     1153  3                 BEGIN
; 1027     1154  3
; 1028     1155  3
; 1029     1156  3                 ! See whether the current record is exhausted. If
; 1030     1157  3                 ! so, get a new record. If none are available,
; 1031     1158  3                 ! return FALSE. Otherwise, set dpc_entry to point to
; 1032     1159  3                 ! the address of the third byte of the correlation record.
; 1033     1160  3                 !
; 1034     1161  4                 IF dpc_entry[current_byte] GTR (.dst_entry[dst$b_length] +
; 1035     1162  4                                 dst_entry[dst$b_length])
; 1036     1163  3                 THEN
; 1037     1164  4                         BEGIN
; 1038     1165  4                         PCTBL_COUNT = .PCTBL_COUNT + 1;
; 1039     1166  4                         IF .PCTBL_COUNT GTR .NUM_PC_TBLS THEN RETURN FALSE;
; 1040     1167  4                         current_table = .current_table + 4;
; 1041     1168  4                         dst_entry = ..current_table;
; 1042     1169  4                         dpc_entry = dst_entry [dst$b_vflags];
; 1043     1170  3                         END;
; 1044     1171  3
; 1045     1172  3
; 1046     1173  3                 ! Now process each command, either PC correlation or
; 1047     1174  3                 ! delta-PC one at a time.
; 1048     1175  3                 !
; 1049     1176  3                 CASE .dpc_entry [current_byte] FROM 1 TO dst$k_pccor_high OF
; 1050     1177  3                         SET
; 1051     1178  3                         [dst$k_delta_pc_w]:
; 1052     1179  4                                 BEGIN
; 1053     1180  4                                 .line_end = (.current_pc - 1) +
; 1054     1181  4                                                 .dpc_entry [next_uns_word];
; 1055     1182  4                                 RETURN TRUE;
```

```
;  1056          1183  3                    END;
;  1057          1184  3
;  1058          1185  3                [dst$k_delta_pc_l]:
;  1059          1186  4                    BEGIN
;  1060          1187  4                    .line_end = (.current_pc - 1) +
;  1061          1188  4                                .dpc_entry [next_uns_long];
;  1062          1189  4                    RETURN TRUE;
;  1063          1190  3                    END;
;  1064          1191  3
;  1065          1192  3                [dst$k_incr_linum]:
;  1066          1193  3                    dpc_entry = dpc_entry [add_two_bytes];
;  1067          1194  3
;  1068          1195  3                [dst$k_incr_linum_w]:
;  1069          1196  3                    dpc_entry = dpc_entry [add_three_bytes];
;  1070          1197  3
;  1071          1198  3                [dst$k_incr_linum_l]:
;  1072          1199  3                    dpc_entry = dpc_entry [add_five_bytes];
;  1073          1200  3
;  1074          1201  3                [dst$k_set_linum_incr]:
;  1075          1202  3                    dpc_entry = dpc_entry [add_two_bytes];
;  1076          1203  3
;  1077          1204  3                [dst$k_set_linum_incr_w]:
;  1078          1205  3                    dpc_entry = dpc_entry [add_three_bytes];
;  1079          1206  3
;  1080          1207  3                [dst$k_reset_linum_incr]:
;  1081          1208  3                    dpc_entry = dpc_entry [add_one_byte];
;  1082          1209  3
;  1083          1210  3                [dst$k_beg_stmt_mode]:
;  1084          1211  3                    dpc_entry = dpc_entry[add_one_byte];
;  1085          1212  3
;  1086          1213  3                [dst$k_end_stmt_mode]:
;  1087          1214  3                    dpc_entry = dpc_entry[add_one_byte];
;  1088          1215  3
;  1089          1216  3                [dst$k_set_linum_b]:
;  1090          1217  3                    dpc_entry = dpc_entry[add_two_bytes];
;  1091          1218  3
;  1092          1219  3                [dst$k_set_linum]:
;  1093          1220  3                    dpc_entry = dpc_entry[add_three_bytes];
;  1094          1221  3
;  1095          1222  3                [dst$k_set_linum_l]:
;  1096          1223  3                    dpc_entry = dpc_entry[add_five_bytes];
;  1097          1224  3
;  1098          1225  3                [dst$k_set_stmtnum]:
;  1099          1226  3                    dpc_entry = dpc_entry[add_three_bytes];
;  1100          1227  3
;  1101          1228  3                [dst$k_set_pc]:
;  1102          1229  4                    BEGIN
;  1103          1230  4                    .line_end = (.start_pc - 1) +
;  1104          1231  4                                .dpc_entry[next_uns_byte];
;  1105          1232  4                    RETURN TRUE;
;  1106          1233  3                    END;
;  1107          1234  3
;  1108          1235  3                [dst$k_set_pc_w]:
;  1109          1236  4                    BEGIN
;  1110          1237  4                    .line_end = (.start_pc - 1) +
;  1111          1238  4                                .dpc_entry[next_uns_word];
;  1112          1239  4                    RETURN TRUE;
```

```
;  1113          1240  3                        END;
;  1114          1241  3
;  1115          1242  3                    [dst$k_set_pc_l]:
;  1116          1243  4                        BEGIN
;  1117          1244  4                        .line_end = (.start_pc - 1) +
;  1118          1245  4                                    .dpc_entry[next_uns_long];
;  1119          1246  4                        RETURN TRUE;
;  1120          1247  3                        END;
;  1121          1248  3
;  1122          1249  3                    [DST$K_SET_ABS_PC]:
;  1123          1250  4                        BEGIN
;  1124          1251  4                        .LINE_END = .DPC_ENTRY[NEXT_UNS_LONG] - 1;
;  1125          1252  4                        RETURN TRUE;
;  1126          1253  3                        END;
;  1127          1254  3
;  1128          1255  3                    [dst$k_term]:
;  1129          1256  4                        BEGIN
;  1130          1257  4                        .line_end = (.current_pc - 1) +
;  1131          1258  4                                    .dpc_entry[next_uns_byte];
;  1132          1259  4                        RETURN TRUE;
;  1133          1260  3                        END;
;  1134          1261  3
;  1135          1262  3                    [dst$k_term_w]:
;  1136          1263  4                        BEGIN
;  1137          1264  4                        .line_end = (.current_pc - 1) +
;  1138          1265  4                                    .dpc_entry[next_uns_word];
;  1139          1266  4                        RETURN TRUE;
;  1140          1267  3                        END;
;  1141          1268  3
;  1142          1269  3                    [dst$k_term_l]:
;  1143          1270  4                        BEGIN
;  1144          1271  4                        .line_end = (.current_pc - 1) +
;  1145          1272  4                                    .dpc_entry[next_uns_long];
;  1146          1273  4                        RETURN TRUE;
;  1147          1274  3                        END;
;  1148          1275  3
;  1149          1276  3                    [OUTRANGE]:
;  1150          1277  4                        BEGIN
;  1151          1278  4                        IF .dpc_entry [current_byte] LSS
;  1152          1279  4                                                dst$k_delta_pc_low
;  1153          1280  4                        OR .dpc_entry[current_byte] GTR
;  1154          1281  4                                                dst$k_delta_pc_high
;  1155          1282  4                        THEN
;  1156          1283  4                                SIGNAL (dbg$_invdstrec);
;  1157          1284  4
;  1158          1285  4                        .line_end = (.current_pc - 1) -
;  1159          1286  4                                    .dpc_entry [current_byte];
;  1160          1287  4                        RETURN TRUE;
;  1161          1288  3                        END;
;  1162          1289  3
;  1163          1290  3                    TES;
;  1164          1291  3
;  1165          1292  2                END;
;  1166          1293  2
;  1167          1294  2            RETURN 0;
;  1168          1295  1            END;
```

```
                                         000C 00000 FIND_EOL:
                                                              .WORD    Save R2,R3                              ; 1126
                        53 00000000'   EF  9E 00002           MOVAB    DPC_ENTRY, R3
                        50        FC   B3  9A 00009 1$:       MOVZBL   @DST_ENTRY, R0                          ; 1162
                        50        FC   A3  C0 0000D           ADDL2    DST_ENTRY, R0
                        50             63  D1 00011           CMPL     DPC_ENTRY, R0                           ; 1161
                                       1B  15 00014           BLEQ     3$
                                  50   A3  D6 00016           INCL     PCTBL_COUNT                             ; 1165
                   38   A3        50   A3  D1 00019           CMPL     PCTBL_COUNT, NUM_PC_TBLS                ; 1166
                                       03  15 0001E           BLEQ     2$                                      ;
                                     00AE  31 00020           BRW      22$                                     ;
                   3C   A3             04  C0 00023 2$:       ADDL2    #4, CURRENT_TABLE                       ; 1167
                   FC   A3        3C   B3  D0 00027           MOVL     @CURRENT_TABLE, DST_ENTRY               ; 1168
              63   FC   A3             02  C1 0002C           ADDL3    #2, DST_ENTRY, DPC_ENTRY                ; 1169
                        52             63  D0 00031 3$:       MOVL     DPC_ENTRY, R2                           ; 1176
              14        01             62  8F 00034           CASEB    (R2), #1, #20                           ;
  0054        005E    0054           0046     00038 4$:       .WORD    6$-4$,-                                 ;
  0050        0050    0050           005E     00040                    9$-4$,-                                 ;
  0074        006A    0064           005E     00048                    11$-4$,-                                ;
  007C        0046    0084           005E     00050                    9$-4$,-                                 ;
  0059        0054    0059           008A     00058                    11$-4$,-                                ;
                                     008A     00060                    8$-4$,-                                 ;
                                                                       8$-4$,-                                 ;
                                                                       8$-4$,-                                 ;
                                                                       11$-4$,-                                ;
                                                                       13$-4$,-                                ;
                                                                       14$-4$,-                                ;
                                                                       16$-4$,-                                ;
                                                                       11$-4$,-                                ;
                                                                       18$-4$,-                                ;
                                                                       6$-4$,-                                 ;
                                                                       17$-4$,-                                ;
                                                                       19$-4$,-                                ;
                                                                       10$-4$,-                                ;
                                                                       9$-4$,-                                 ;
                                                                       10$-4$,-                                ;
                                                                       19$-4$                                  ;
                                       62  95 00062           TSTB     (R2)                                    ; 1280
                                       0D  15 00064           BLEQ     5$                                      ;
                           0002832A    8F  DD 00066           PUSHL    #164650                                 ; 1283
                00000000G   00         01  FB 0006C           CALLS    #1, LIB$SIGNAL                          ;
                            50    00   B3  98 00073 5$:       CVTBL    @DPC_ENTRY, R0                          ; 1286
              50       14   A3         50  C3 00077           SUBL3    R0, CURRENT_PC, R0                      ;
                                       4A  11 0007C           BRB      20$                                     ; 1285
                            50    01   A2  3C 0007E 6$:       MOVZWL   1(R2), R0                               ; 1181
                            50    14   A3  C0 00082 7$:       ADDL2    CURRENT_PC, R0                          ;
                                       40  11 00086           BRB      20$                                     ; 1180
                                       63  D6 00088 8$:       INCL     DPC_ENTRY                               ; 1214
                                       0D  11 0008A           BRB      12$                                     ;
                            63         02  C0 0008C 9$:       ADDL2    #2, DPC_ENTRY                           ; 1217
                                       08  11 0008F           BRB      12$                                     ;
                            63         05  C0 00091 10$:      ADDL2    #5, DPC_ENTRY                           ; 1223
                                       03  11 00094           BRB      12$                                     ;
```

```
                              63          03  C0 00096 11$:     ADDL2   #3, DPC_ENTRY                        ; 1226
                                        FF6D  31 00099 12$:     BRW     1$                                   ;
                              50      01  A2  9A 0009C 13$:     MOVZBL  1(R2), R0                            ; 1231
                                          04  11 000A0          BRB     15$                                  ;
                              50      01  A2  3C 000A2 14$:     MOVZWL  1(R2), R0                            ; 1238
                              50      04  A3  C0 000A6 15$:     ADDL2   START_PC, R0                         ;
                                          1C  11 000AA          BRB     20$                                  ; 1237
              50        04    A3      01  A2  C1 000AC 16$:     ADDL3   1(R2), START_PC, R0                  ; 1245
                                          14  11 000B2          BRB     20$                                  ; 1244
        04    BC        01    A2          01  C3 000B4 17$:     SUBL3   #1, 1(R2), @LINE_END                 ; 1251
                                          11  11 000BA          BRB     21$                                  ; 1252
                              50      01  A2  9A 000BC 18$:     MOVZBL  1(R2), R0                            ; 1258
                                          C0  11 000C0          BRB     7$                                   ; 1265
              50        14    A3      01  A2  C1 000C2 19$:     ADDL3   1(R2), CURRENT_PC, R0                ; 1272
                        04    BC      FF  A0  9E 000C8 20$:     MOVAB   -1(R0), @LINE_END                    ; 1271
                              50          01  D0 000CD 21$:     MOVL    #1, R0                               ; 1273
                                              04 000D0          RET                                          ;
                                          50  D4 000D1 22$:     CLRL    R0                                   ; 1295
                                              04 000D3          RET                                          ;

; Routine Size:  212 bytes,    Routine Base:  DBG$CODE + 04C3
```

```
;  1170          1296  1 ROUTINE GIVE_LINE_INFO(LINE_NUM, STMT_NUM): NOVALUE =
;  1171          1297  1 !
;  1172          1298  1 ! FUNCTION
;  1173          1299  1 !       This routine gives prev., current, next line information to the user
;  1174          1300  1 !       when the desired line is not found.
;  1175          1301  1 !
;  1176          1302  1 ! INPUTS
;  1177          1303  1 !       REPORT_PREV_LINE - Previous line
;  1178          1304  1 !       REPORT_PREV_STMT - Previous statement
;  1179          1305  1 !       LINE_NUM         - Current line
;  1180          1306  1 !       STMT_NUM         - Current statement
;  1181          1307  1 !       REPORT_NEXT_LINE - Next line
;  1182          1308  1 !       REPORT_NEXT_STMT - Next statement
;  1183          1309  1 !
;  1184          1310  1 ! OUTPUTS
;  1185          1311  1 !       Informational message is displayed.  No return value.
;  1186          1312  1 !
;  1187          1313  1
;  1188          1314  2     BEGIN
;  1189          1315  2
;  1190          1316  2     LOCAL
;  1191          1317  2         BUFFER: VECTOR[80, BYTE],       ! Output buffer
;  1192          1318  2         BUF_DESC: VECTOR[2, LONG];      ! Output buffer string descriptor
;  1193          1319  2
;  1194          1320  2
;  1195          1321  2     IF .STMT_NUM EQL 0 THEN STMT_NUM = 1;
;  1196          1322  2     IF .REPORT_PREV_STMT EQL 0 THEN REPORT_PREV_STMT = 1;
;  1197          1323  2     IF .REPORT_NEXT_STMT EQL 0 THEN REPORT_NEXT_STMT = 1;
;  1198          1324  2
;  1199          1325  2     BUF_DESC[0] = 79;
;  1200          1326  2     BUF_DESC[1] = BUFFER[1];
;  1201          1327  2
;  1202          1328  2     IF (.REPORT_PREV_LINE EQL 0) AND
;  1203          1329  2        (.LINE_NUM EQL .REPORT_NEXT_LINE) AND
;  1204          1330  2        (.REPORT_PREV_STMT EQL 1) AND
;  1205          1331  2        (.STMT_NUM EQL .REPORT_NEXT_STMT)
;  1206          1332  2     THEN
;  1207          1333  3         BEGIN
;  1208          1334  3         DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE
;  1209          1335  3             (%ASCIC 'no line information available'));
;  1210          1336  3         BUFFER[0] = 79 - .BUF_DESC[0];
;  1211          1337  3         SIGNAL(DBG$_LINEINFO, 1, BUFFER);
;  1212          1338  3         RETURN 0;
;  1213          1339  3         END;
;  1214          1340  2
;  1215          1341  2     DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC 'no line !UL'), .LINE_NUM);
;  1216          1342  2     IF .STMT_NUM GTR 1
;  1217          1343  2     THEN
;  1218          1344  2         DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC '.!UL'), .STMT_NUM);
;  1219          1345  2
;  1220          1346  2     IF NOT (.REPORT_PREV_LINE EQL 0 AND .REPORT_PREV_STMT EQL 1)
;  1221          1347  2     THEN
;  1222          1348  3         BEGIN
;  1223          1349  3         DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE
;  1224          1350  3                 (%ASCIC ', previous line is !UL'), .REPORT_PREV_LINE);
;  1225          1351  3
;  1226          1352  3         IF .REPORT_PREV_STMT GTR 1
```

```
3             THEN
                  DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC '.!UL'), .REPORT_PREV_STMT);
2             END;
2
3         IF NOT (.REPORT_NEXT_LINE EQL .LINE_NUM  AND
3                 .REPORT_NEXT_STMT EQL .STMT_NUM)
2         THEN
3             BEGIN
3             DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE
3                     (%ASCIC ', next line is !UL'), .REPORT_NEXT_LINE);
3
3             IF .REPORT_NEXT_STMT GTR 1
3             THEN
3                 DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC '.!UL'), .REPORT_NEXT_STMT);
2             END;
2
2         BUFFER[0] = 79 - .BUF_DESC[0];
2         SIGNAL(DBG$_LINEINFO, 1, BUFFER);
2         RETURN 0;
1         END;


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

6D 72 6F 66 6E 69 20 65 6E 69 6C 20 6F 6E 1D 00000 P.AAA:  .ASCII  <29>\no line information available\   ;
65 6C 62 61 6C 69 61 76 61 20 6E 6F 69 74 61 0000F                                                     ;
         4C 55 21 20 65 6E 69 6C 20 6F 6E 0B 0001E P.AAB:  .ASCII  <11>\no line !UL\                    ;
                              4C 55 21 2E 04 0002A P.AAC:  .ASCII  <4>\.!UL\                            ;
6E 69 6C 20 73 75 6F 69 76 65 72 70 20 2C 16 0002F P.AAD:  .ASCII  <22>\, previous line is !UL\         ;
                     4C 55 21 20 73 69 20 65 0003E                                                      ;
                              4C 55 21 2E 04 00046 P.AAE:  .ASCII  <4>\.!UL\                            ;
73 69 20 65 6E 69 6C 20 74 78 65 6E 20 2C 12 0004B P.AAF:  .ASCII  <18>\, next line is !UL\             ;
                                 4C 55 21 20 0005A                                                      ;
                              4C 55 21 2E 04 0005E P.AAG:  .ASCII  <4>\.!UL\                            ;


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

                                   001C 00000 GIVE_LINE_INFO:
                                                           .WORD   Save R2,R3,R4                        ; 1296
                 54 00000000G  00  9E 00002                MOVAB   DBG$FORMAT_FAO_OUT, R4               ;
                 53 00000000'  EF  9E 00009                MOVAB   P.AAA, R3                            ;
                 52 00000000'  EF  9E 00010                MOVAB   REPORT_PREV_STMT, R2                 ;
                 5E         A8 AE  9E 00017                MOVAB   -88(SP), SP                          ;
                            08 AC  D5 0001B                TSTL    STMT_NUM                             ; 1321
                               04  12 0001E                BNEQ    1$                                   ;
            08   AC            01  D0 00020                MOVL    #1, STMT_NUM                         ;
                               62  D5 00024 1$:            TSTL    REPORT_PREV_STMT                     ; 1322
                               03  12 00026                BNEQ    2$                                   ;
                 62            01  D0 00028                MOVL    #1, REPORT_PREV_STMT                 ;
                            F8 A2  D5 0002B 2$:            TSTL    REPORT_NEXT_STMT                     ; 1323
                               04  12 0002E                BNEQ    3$                                   ;
            F8   A2            01  D0 00030                MOVL    #1, REPORT_NEXT_STMT                 ;
                 6E         4F 8F  9A 00034 3$:            MOVZBL  #79, BUF_DESC                        ; 1325
            04   AE         09 AE  9E 00038                MOVAB   BUFFER+1, BUF_DESC+4                 ; 1326
```

```
                        FC   A2   D5 0003D          TSTL    REPORT_PREV_LINE                                ; 1328
                             1D   12 00040          BNEQ    4$
                  F4    A2       04 AC   D1 00042          CMPL    LINE_NUM, REPORT_NEXT_LINE              ; 1329
                                   16   12 00047          BNEQ    4$
                        01         62   D1 00049          CMPL    REPORT_PREV_STMT, #1                    ; 1330
                                   11   12 0004C          BNEQ    4$
                  F8    A2       08 AC   D1 0004E          CMPL    STMT_NUM, REPORT_NEXT_STMT              ; 1331
                                   0A   12 00053          BNEQ    4$
                                   53   DD 00055          PUSHL   R3                                      ; 1334
                                   04 AE   9F 00057          PUSHAB  BUF_DESC
                        64         02   FB 0005A          CALLS   #2, DBG$FORMAT_FAO_OUT
                                   70   11 0005D          BRB     9$                                      ; 1336
                                   04 AC   DD 0005F  4$:     PUSHL   LINE_NUM                                ; 1341
                                   1E A3   9F 00062          PUSHAB  P.AAB
                                   08 AE   9F 00065          PUSHAB  BUF_DESC
                        64         03   FB 00068          CALLS   #3, DBG$FORMAT_FAO_OUT
                        01       08 AC   D1 0006B          CMPL    STMT_NUM, #1                            ; 1342
                                   0C   15 0006F          BLEQ    5$
                                   08 AC   DD 00071          PUSHL   STMT_NUM                                ; 1344
                                   2A A3   9F 00074          PUSHAB  P.AAC
                                   08 AE   9F 00077          PUSHAB  BUF_DESC
                        64         03   FB 0007A          CALLS   #3, DBG$FORMAT_FAO_OUT
                        50       FC A2   D0 0007D  5$:     MOVL    REPORT_PREV_LINE, R0                    ; 1346
                                   05   12 00081          BNEQ    6$
                        01         62   D1 00083          CMPL    REPORT_PREV_STMT, #1
                                   1B   13 00086          BEQL    7$
                                   50   DD 00088  6$:     PUSHL   R0                                      ; 1350
                                   2F A3   9F 0008A          PUSHAB  P.AAD                                   ; 1349
                                   08 AE   9F 0008D          PUSHAB  BUF_DESC
                        64         03   FB 00090          CALLS   #3, DBG$FORMAT_FAO_OUT
                        01         62   D1 00093          CMPL    REPORT_PREV_STMT, #1                    ; 1352
                                   08   15 00096          BLEQ    7$
                                   62   DD 00098          PUSHL   REPORT_PREV_STMT                        ; 1354
                                   46 A3   9F 0009A          PUSHAB  P.AAE
                                   08 AE   9F 0009D          PUSHAB  BUF_DESC
                        64         03   FB 000A0          CALLS   #3, DBG$FORMAT_FAO_OUT
                  04    AC       F4 A2   D1 000A3  7$:     CMPL    REPORT_NEXT_LINE, LINE_NUM              ; 1357
                                   07   12 000A8          BNEQ    8$
                  08    AC       F8 A2   D1 000AA          CMPL    REPORT_NEXT_STMT, STMT_NUM              ; 1358
                                   1E   13 000AF          BEQL    9$
                                   F4 A2   DD 000B1  8$:     PUSHL   REPORT_NEXT_LINE                        ; 1362
                                   4B A3   9F 000B4          PUSHAB  P.AAF                                   ; 1361
                                   08 AE   9F 000B7          PUSHAB  BUF_DESC
                        64         03   FB 000BA          CALLS   #3, DBG$FORMAT_FAO_OUT
                        01       F8 A2   D1 000BD          CMPL    REPORT_NEXT_STMT, #1                    ; 1364
                                   0C   15 000C1          BLEQ    9$
                                   F8 A2   DD 000C3          PUSHL   REPORT_NEXT_STMT                        ; 1366
                                   5E A3   9F 000C6          PUSHAB  P.AAG
                                   08 AE   9F 000C9          PUSHAB  BUF_DESC
                        64         03   FB 000CC          CALLS   #3, DBG$FORMAT_FAO_OUT
08    AE        4F    8F               6E   83 000CF  9$:     SUBB3   BUF_DESC, #79, BUFFER                   ; 1369
                                   08 AE   9F 000D5          PUSHAB  BUFFER                                  ; 1370
                                   01   DD 000D8          PUSHL   #1
                        00028703   8F   DD 000DA          PUSHL   #165635
              00000000G 00               03   FB 000E0          CALLS   #3, LIB$SIGNAL
                                         04   000E7          RET                                             ; 1372
```

```
; Routine Size:  232 bytes,    Routine Base:  DBG$CODE + 0597

; 1247          1373  1
; 1248          1374  1 END
; 1249          1375  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
;  DBG$OWN                        88  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                     1663  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$PLIT                       99  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         0         0        1000       00:01.8
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         0         0           7       00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1     1545        56         3          97       00:01.8
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418       127        30          31       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386         2         0          22       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1      150         0         0          12       00:00.3



;                               COMMAND QUALIFIERS

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

; Size:          1663 code + 187 data bytes
; Run Time:         00:35.2
; Elapsed Time:     02:05.1
; Lines/CPU Min:     2343
; Lexemes/CPU-Min: 12071
; Memory Used:   221 pages
; Compilation Complete
```

DBGCVTMAC
LIS

DBGDPC
LIS

DBGDEFINE
LIS

0080 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGEVALOP
LIS

DBGENCDEC
LIS