```
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  MM      MM    OOOOOO    DDDDDDDD
DDDDDDDD    BBBBBBBB      GGGGGGGG  MM      MM    OOOOOO    DDDDDDDD
DD      DD  BB      BB  GG          MMMM  MMMM  OO      OO  DD      DD
DD      DD  BB      BB  GG          MMMM  MMMM  OO      OO  DD      DD
DD      DD  BB      BB  GG          MM  MM  MM  OO      OO  DD      DD
DD      DD  BB      BB  GG          MM  MM  MM  OO      OO  DD      DD
DD      DD  BBBBBBBB    GG          MM      MM  OO      OO  DD      DD
DD      DD  BBBBBBBB    GG          MM      MM  OO      OO  DD      DD
DD      DD  BB      BB  GG  GGGGGG  MM      MM  OO      OO  DD      DD
DD      DD  BB      BB  GG  GGGGGG  MM      MM  OO      OO  DD      DD
DD      DD  BB      BB  GG      GG  MM      MM  OO      OO  DD      DD      ....
DD      DD  BB      BB  GG      GG  MM      MM  OO      OO  DD      DD      ....
DDDDDDDD    BBBBBBBB      GGGGGG    MM      MM    OOOOOO    DDDDDDDD      ....
DDDDDDDD    BBBBBBBB      GGGGGG    MM      MM    OOOOOO    DDDDDDDD      ....


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 DBGMOD (IDENT = 'V04-000') =
;     2          0002  1 BEGIN
;     3          0003  1
;     4          0004  1 !**************************************************************************
;     5          0005  1 !*                                                                        *
;     6          0006  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     7          0007  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     8          0008  1 !*  ALL RIGHTS RESERVED.                                                  *
;     9          0009  1 !*                                                                        *
;    10          0010  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    11          0011  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    12          0012  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    13          0013  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    14          0014  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    15          0015  1 !*  TRANSFERRED.                                                           *
;    16          0016  1 !*                                                                        *
;    17          0017  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    18          0018  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    19          0019  1 !*  CORPORATION.                                                           *
;    20          0020  1 !*                                                                        *
;    21          0021  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    22          0022  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    23          0023  1 !*                                                                        *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !**************************************************************************
;    26          0026  1
;    27          0027  1 ! FACILITY:     DEBUG
;    28          0028  1 !
;    29          0029  1 !++
;    30          0030  1 ! FUNCTIONAL DESCRIPTION:
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 ! Version:      1.08
;    35          0035  1 !
;    36          0036  1 ! History:
;    37          0037  1 !       Author:
;    38          0038  1 !               Carol Peters, 03 Jul 1976: Version 01
;    39          0039  1 !
;    40          0040  1 !       Modified by:
;    41          0041  1 !               Mike Candela, 18 October 1979
;    42          0042  1 !               Richard Title, 12 AUG 1981
;    43          0043  1 !
;    44          0044  1 ! Revision history:
;    45          0045  1 ! 3.00  21-Aug-81       RT      Added support for SET STEP SOURCE and
;    46          0046  1 !                               SET STEP NOSOURCE
;    47          0047  1 ! 3.01  20-Oct-81       RT      Added support for SEARCH command.
;    48          0048  1 ! 3.02  06-May-82       RT      Added support for SET DEFINE and SHOW DEFINE
;    49          0049  1 !       17-Aug-83       PS      Added mode G/NOG_FLOAT
;    50          0050  1 !--
```

```
;    52          0051  1 ! TABLE OF CONTENTS:
;    53          0052  1 !
;    54          0053  1 FORWARD ROUTINE
;    55          0054  1         dbg$init_modes:  NOVALUE,               ! SETS MODES TO A SPECIFIED level
;    56          0055  1         dbg$set_mod_lvl: NOVALUE,
;    57          0056  1         dbg$set_mod_def: NOVALUE,               ! SETS DEFAULT MODES TO INITIALIZATION values
;    58          0057  1         dbg$show_mode:   NOVALUE,               ! ACTION ROUTINE TO SHOW DEFAULT modes
;    59          0058  1         dbg$init_step:   NOVALUE,               ! SETS step LEVELS AS SPECIFIED
;    60          0059  1         dbg$set_stp_def: NOVALUE,               ! RESETS step LEVELS TO INITIALIZATION VALUES
;    61          0060  1         dbg$set_stp_lvl: NOVALUE,               ! SETS level OF step
;    62          0061  1         dbg$show_step:   NOVALUE,               ! ACTION ROUTINE TO SHOW step
;    63          0062  1         dbg$init_search: NOVALUE,               ! resets search levels to initial
;    64          0063  1         dbg$set_search_def: NOVALUE,            ! sets search levels as specified
;    65          0064  1         dbg$set_search_lvl: NOVALUE,            ! set level of search
;    66          0065  1         dbg$show_search: NOVALUE,               ! action routine for show search
;    67          0066  1         dbg$init_define: NOVALUE,               ! resets define levels to initial
;    68          0067  1         dbg$set_define_def: NOVALUE,            ! sets define levels as specified
;    69          0068  1         dbg$set_define_lvl: NOVALUE,            ! set level of define
;    70          0069  1         dbg$show_define: NOVALUE,               ! action routine for show define
;    71          0070  1         dbg$set_out_def: NOVALUE,               ! Sets default OUTPUT config
;    72          0071  1         dbg$show_output: NOVALUE,               ! Displays OUTPUT configuration
;    73          0072  1         dbg$set_log:     NOVALUE,               ! Set a new LOG file name
;    74          0073  1         setup_log:       ,                      ! Creates & opens log file
;    75          0074  1         dbg$verify_out:  NOVALUE;               ! Decides whether to VERIFY input
;    76          0075  1 !
;    77          0076  1 ! REQUIRE FILES:
;    78          0077  1 !
;    79          0078  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    80          0212  1 LIBRARY 'LIB$:DBGGEN.L32';
;    81          0213  1
;    82          0214  1 EXTERNAL ROUTINE
;    83          0215  1         dbg$fao_out: NOVALUE,
;    84          0216  1         dbg$get_memory,                         ! allocate block of free storage
;    85          0217  1         dbg$newline: NOVALUE,                   ! output a line
;    86          0218  1         dbg$nget_trans_radix,                   ! returns translation of default radix
;    87          0219  1         DBG$CONV_R_50,                          ! converts from radix 50
;    88          0220  1         dbg$print: NOVALUE,                     ! put a line in the output buffer
;    89          0221  1         dbg$rel_memory : NOVALUE,               ! return storage to free pool
;    90          0222  1         dbg$show_radix: NOVALUE;
;    91          0223  1
;    92          0224  1 EXTERNAL
;    93          0225  1         DBG$GB_KEYPAD_INPUT: BYTE,
;    94          0226  1         DBG$GL_SCREEN_MODE,
;    95          0227  1         DBG$GB_DEF_STP :    BLOCKVECTOR
;    96          0228  1                             [STEP_LEVELS, EVENT$K_STEPPING_DESC_SIZE]
;    97          0229  1                             FIELD (EVENT$STEPPING_DESC_FIELDS),
;    98          0230  1         DBG$GB_STP_PTR :    REF EVENT$STEPPING_DESCRIPTOR, ! current stepping
;    99          0231  1         dbg$gb_def_search: VECTOR [, BYTE],     ! Block for SEARCH settings
;   100          0232  1         dbg$gb_search_ptr: REF VECTOR [, BYTE], ! pointer to search settings
;   101          0233  1         dbg$gb_def_define: VECTOR [, BYTE],     ! Block for DEFINE settings
;   102          0234  1         dbg$gb_define_ptr: REF VECTOR [, BYTE], ! pointer to define settings
;   103          0235  1         dbg$gb_def_mod: VECTOR [, BYTE],        ! BLOCK FOR MODES
;   104          0236  1         dbg$gb_def_out: VECTOR [, BYTE],        ! Vector for output config
;   105          0237  1         dbg$gl_cishead: REF cis$link,           ! Head of cis
;   106          0238  1         dbg$gb_verptr : REF VECTOR [,BYTE],     ! Verify pointer
;   107          0239  1         dbg$gb_mod_ptr: REF VECTOR [, BYTE],    ! pointer TO MODES
;   108          0240  1         dbg$gl_context : BITVECTOR,             ! CONTEXT WORD
```

```
;  109          0241  1          dbg$gb_language: BYTE,                          ! contains index to current language
;  110          0242  1          dbg$gl_lognam : REF $NAM_DECL,                  ! NAM block for LOG file
;  111          0243  1          dbg$gl_logfab : BLOCK [,BYTE],                  ! FAB for LOG file
;  112          0244  1          dbg$gl_lograb : BLOCK [,BYTE],                  ! RAB for LOG file
;  113          0245  1          dbg$gb_logfsr : REF VECTOR [,BYTE],             ! Resultant LOG filespec buffer
;  114          0246  1          dbg$gb_logfse : REF VECTOR [,BYTE],             ! Expanded LOG filespec buffer
;  115          0247  1          dbg$gl_log_buf,                                 ! pointer to buffer containing
;  116          0248  1                                                          ! filespec from SET LOG cmd.
;  117          0249  1          DBG$OPCODE_KIND_TABLE: VECTOR[512,WORD],
;  118          0250  1          DBG$OPCODE_NAME_TABLE: BLOCKVECTOR[,10,BYTE];
;  119          0251  1
;  120          0252  1 LITERAL
;  121          0253  1          dbg_mod         = 0,                            ! DEBUGGING AID FOR mode
;  122          0254  1          dbg_mod1        = 0,                            ! DEBUGGING AID FOR step
;  123          0255  1          dbg_mod2        = 0;                            !                  "   "   log file
;  124          0256  1
;  125          0257  1 !++
;  126          0258  1 ! The following table holds the valid mode settings and values
;  127          0259  1 ! relevant to them. Each entry in the table is three bytes long.
;  128          0260  1 ! The entry has three fields, each of them one byte
;  129          0261  1 ! long.
;  130          0262  1 !
;  131          0263  1 !       ------------------------------
;  132          0264  1 !       ! keyword !  offset ! value !
;  133          0265  1 !       ------------------------------
;  134          0266  1 !
;  135          0267  1 ! The keyword field holds the token value that represents the particular
;  136          0268  1 ! mode, e.g., byte_token indicates that the mode is BYTE.
;  137          0269  1 ! The offset field holds the location of the mode setting in any of
;  138          0270  1 ! the mode levels.
;  139          0271  1 ! The value field holds the value that represents a particular mode
;  140          0272  1 ! to DEBUG in an internal sense.
;  141          0273  1 !
;  142          0274  1 LITERAL
;  143          0275  1          keyword_field   = 0,
;  144          0276  1          offset_field    = 1,
;  145          0277  1          value_field     = 2,
;  146          0278  1          mode_entry_len  = 3,
;  147          0279  1          step_entry_len  = 3,
;  148          0280  1          search_entry_len = 3;
;  149          0281  1
;  150          0282  1 BIND
;  151          0283  1          deflog_name = UPLIT BYTE(%ASCII 'DEBUG.LOG'),
;  152          0284  1          deflog_size = %CHARCOUNT(%ASCII 'DEBUG.LOG');
;  153          0285  1
;  154          0286  1 ! build_err_desc creates a descriptor that contains the filespec for RMS error
;  155          0287  1 ! reporting using the shared message file
;  156          0288  1
;  157          0289  1 MACRO
;  158        M 0290  1          build_err_desc (msg_desc) =
;  159        M 0291  1
;  160        M 0292  1          IF .dbg$gl_log_buf NEQ 0
;  161        M 0293  1          THEN
;  162        M 0294  1                  BEGIN
;  163        M 0295  1                  msg_desc [dsc$w_length]  = .dbg$gl_logfab [fab$b_fns];
;  164        M 0296  1                  msg_desc [dsc$a_pointer] = .dbg$gl_logfab [fab$l_fna];
;  165        M 0297  1                  END
```

```
;  166        M 0298  1          ELSE
;  167        M 0299  1                  BEGIN
;  168        M 0300  1                  msg_desc [dsc$w_length]  = .dbg$gl_logfab [fab$b_dns];
;  169        M 0301  1                  msg_desc [dsc$a_pointer] = .dbg$gl_logfab [fab$l_dna];
;  170          0302  1                  END %,
;  171          0303  1
;  172          0304  1 ! signal_rms_err reports RMS errors caused by $CREATE, $OPEN or $CONNECT
;  173          0305  1
;  174        M 0306  1          signal_rms_err (block_type) =
;  175        M 0307  1                  BEGIN
;  176        M 0308  1                  LOCAL
;  177        M 0309  1                          msg_desc : BLOCK [8,BYTE],
;  178        M 0310  1                          sts,
;  179        M 0311  1                          stv ;
;  180        M 0312  1
;  181        M 0313  1                  build_err_desc (msg_desc) ;
;  182        M 0314  1                  IF block_type EQL 1
;  183        M 0315  1                  THEN
;  184        M 0316  1                          BEGIN
;  185        M 0317  1                          sts = .dbg$gl_logfab [fab$l_sts];
;  186        M 0318  1                          stv = .dbg$gl_logfab [fab$l_stv];
;  187        M 0319  1                          END
;  188        M 0320  1                  ELSE
;  189        M 0321  1                          BEGIN
;  190        M 0322  1                          sts = .dbg$gl_lograb [rab$l_sts];
;  191        M 0323  1                          stv = .dbg$gl_lograb [rab$l_stv];
;  192        M 0324  1                          END;
;  193        M 0325  1                  SIGNAL (shr$_openout+dbg_fac_code, 1, msg_desc, .sts, .stv);
;  194          0326  1                  END %,
;  195          0327  1
;  196          0328  1 ! restore_nam restores an RMS NAM block
;  197          0329  1
;  198        M 0330  1          restore_nam =
;  199        M 0331  1                  BEGIN
;  200        M 0332  1                  LOCAL
;  201        M 0333  1                          err,
;  202        M 0334  1                          type_b;
;  203        M 0335  1
;  204        M 0336  1                  ! release storage for the aborted log file
;  205        M 0337  1
;  206        M 0338  1                  dbg$rel_memory ( .temp_nam);
;  207        M 0339  1                  dbg$rel_memory ( .temp_fsr);
;  208        M 0340  1                  dbg$rel_memory ( .temp_fse);
;  209        M 0341  1                  dbg$gl_lognam = .old_nam_ptr;
;  210        M 0342  1                  old_nam_ptr = 0;
;  211        M 0343  1                  dbg$gl_logfab [fab$l_nam] = .dbg$gl_lognam;
;  212        M 0344  1                  dbg$gl_logfab [fab$l_fna] = .fna;
;  213        M 0345  1                  dbg$gl_logfab [fab$b_fns] = .fns;
;  214        M 0346  1
;  215        M 0347  1                  dbg$gl_logfab [fab$v_nam] = 1;
;  216        M 0348  1                  dbg$gl_logfab [fab$v_cif] = 1;
;  217        M 0349  1                  dbg$gl_logfab [fab$v_mxv] = 0;
;  218        M 0350  1                  err = setup_log(type_b);
;  219        M 0351  1                  IF NOT .err
;  220        M 0352  1                  THEN
;  221        M 0353  1                          signal_rms_err (.type_b) ;
;  222        M 0354  1
```

```
;  223        M 0355  1          dbg$gb_def_out [out_log] = .log_temp ;
;  224          0356  1          END%;
```

```
;  226          0357  1 GLOBAL ROUTINE dbg$init_modes (goal_level, source_level) : NOVALUE =
;  227          0358  1 !++
;  228          0359  1 ! FUNCTIONAL DESCRIPTION:
;  229          0360  1 !       Sets all levels from local level to the goal level specified
;  230          0361  1 !       with the mode settings of the source level.
;  231          0362  1 !
;  232          0363  1 ! FORMAL PARAMETERS:
;  233          0364  1 !       goal_level      - highest level to set
;  234          0365  1 !       source_level    - level from which to obtain mode settings
;  235          0366  1 !
;  236          0367  1 ! IMPLICIT INPUTS:
;  237          0368  1 !       none
;  238          0369  1 !
;  239          0370  1 ! IMPLICIT OUTPUTS:
;  240          0371  1 !       THE mode settings of the local level to the goal level
;  241          0372  1 !       are reset to the mode settings of the source level.
;  242          0373  1 !
;  243          0374  1 ! Routine value:
;  244          0375  1 !       NOVALUE
;  245          0376  1 !
;  246          0377  1 ! SIDE EFFECTS:
;  247          0378  1 !       NONE
;  248          0379  1 !--
;  249          0380  1
;  250          0381  2         BEGIN
;  251          0382  2
;  252          0383  2         LOCAL
;  253          0384  2                 temp_level: REF VECTOR [, BYTE];
;  254          0385  2
;  255        L 0386  2 %IF dbg_mod
;  256        U 0387  2 %THEN
;  257        U 0388  2         $fao_tt_out ('copying level !SL modes to level !SL thru !SL',
;  258        U 0389  2                         .source_level, .goal_level, local_mode);
;  259          0390  2 %FI
;  260          0391  2
;  261          0392  2         temp_level = dbg$gb_def_mod [.source_level * mode_lvl_size];
;  262          0393  2         DECR I FROM local_mode TO .goal_level DO
;  263          0394  3                 BEGIN
;  264          0395  3                 dbg$gb_mod_ptr = dbg$gb_def_mod [.I * mode_lvl_size];
;  265          0396  3                 dbg$gb_mod_ptr [mode_radix] = .temp_level [mode_radix];
;  266          0397  3                 dbg$gb_mod_ptr [mode_length] = .temp_level [mode_length];
;  267          0398  3                 dbg$gb_mod_ptr [mode_symbols] = .temp_level [mode_symbols];
;  268          0399  3                 dbg$gb_mod_ptr [mode_g_floats] = .temp_level [mode_g_floats];
;  269          0400  3                 dbg$gb_mod_ptr [mode_instruc] = .temp_level [mode_instruc];
;  270          0401  3                 dbg$gb_mod_ptr [mode_ascii] = .temp_level [mode_ascii];
;  271          0402  3                 dbg$gb_mod_ptr [mode_scope] = .temp_level [mode_scope];
;  272          0403  3                 dbg$gb_mod_ptr [mode_globals] = .temp_level [mode_globals];
;  273          0404  3                 dbg$gb_mod_ptr [mode_immediate] = .temp_level [mode_immediate];
;  274          0405  3                 dbg$gb_mod_ptr [mode_fortran] = .temp_level [mode_fortran];
;  275          0406  2                 END;
;  276          0407  1         END;


                                                        .TITLE  DBGMOD
                                                        .IDENT  \V04-000\

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

```
                47 4F 4C 2E 47 55 42 45 44  00000 P.AAA:  .ASCII  \DEBUG.LOG\                                    ;

                                                    DEFLOG_NAME=        P.AAA
                                                    DEFLOG_SIZE=        9
                                                            .EXTRN  DBG$FAO_OUT, DBG$GET_MEMORY
                                                            .EXTRN  DBG$NEWLINE, DBG$NGET_TRANS_RADIX
                                                            .EXTRN  DBG$CONV_R_50, DBG$PRINT
                                                            .EXTRN  DBG$REL_MEMORY, DBG$SHOW_RADIX
                                                            .EXTRN  DBG$GB_KEYPAD_INPUT
                                                            .EXTRN  DBG$GL_SCREEN_MODE
                                                            .EXTRN  DBG$GB_DEF_STP, DBG$GB_STP_PTR
                                                            .EXTRN  DBG$GB_DEF_SEARCH
                                                            .EXTRN  DBG$GB_SEARCH_PTR
                                                            .EXTRN  DBG$GB_DEF_DEFINE
                                                            .EXTRN  DBG$GB_DEFINE_PTR
                                                            .EXTRN  DBG$GB_DEF_MOD, DBG$GB_DEF_OUT
                                                            .EXTRN  DBG$GL_CISREAD, DBG$GB_VERPTR
                                                            .EXTRN  DBG$GB_MOD_PTR, DBG$GL_CONTEXT
                                                            .EXTRN  DBG$GB_LANGUAGE
                                                            .EXTRN  DBG$GL_LOGNAM, DBG$GL_LOGFAB
                                                            .EXTRN  DBG$GL_LOGRAB, DBG$GB_LOGFSR
                                                            .EXTRN  DBG$GB_LOGFSE, DBG$GL_LOG_BUF
                                                            .EXTRN  DBG$OPCODE_KIND_TABLE
                                                            .EXTRN  DBG$OPCODE_NAME_TABLE

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

                                                  001C 00000        .ENTRY  DBG$INIT_MODES, Save R2,R3,R4                   ; 0357
                          54 00000000G   00  9E 00002        MOVAB   DBG$GB_DEF_MOD, R4                              ;
                          53 00000000G   00  9E 00009        MOVAB   DBG$GB_MOD_PTR, R3                              ;
                50    08  AC             0A  C5 00010        MULL3   #10, SOURCE_LEVEL, R0                           ; 0392
                52        50             54  C1 00015        ADDL3   R4, R0, TEMP_LEVEL                              ;
                          50             03  D0 00019        MOVL    #3, I                                           ; 0405
                                         1A  11 0001C        BRB     2$                                              ;
                51        50             0A  C5 0001E 1$:    MULL3   #10, I, R1                                      ; 0395
                63        51             54  C1 00022        ADDL3   R4, R1, DBG$GB_MOD_PTR                          ;
                          51             63  D0 00026        MOVL    DBG$GB_MOD_PTR, R1                              ; 0396
                          61             62  D0 00029        MOVL    (TEMP_LEVEL), (R1)                              ;
                    04    A1       04    A2  B0 0002C        MOVW    4(TEMP_LEVEL), 4(R1)                            ; 0401
                    06    A1       06    A2  D0 00031        MOVL    6(TEMP_LEVEL), 6(R1)                            ; 0403
                                         50  D7 00036        DECL    I                                               ; 0393
                    04    AC             50  D1 00038 2$:    CMPL    I, GOAL_LEVEL                                   ;
                                         E0  18 0003C        BGEQ    1$                                              ;
                                             04 0003E        RET                                                     ; 0407

; Routine Size:  63 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  278          0408  1 !++
;  279          0409  1 ! THE FOLLOWING ROUTINES CONTROL INTERPRETATION AND TYPEOUT MODES.
;  280          0410  1 !
;  281          0411  1 !       ADDRESS INTERPRETATION AND DATA TYPEOUT ARE CONTROLLED BY FOUR MODE
;  282          0412  1 !       SETTINGS. THEY ARE 'DEFAULT_MODE', WHICH IS THE MODE SET BY
;  283          0413  1 !       DEBUG INITIALIZATION; 'user_def_mode', WHICH IS THE MODE SET
;  284          0414  1 !       BY USER 'SET DEFAULT' COMMANDS; 'override_mode', WHICH IS
;  285          0415  1 !       THE MODE SET BY A SINGLE LINE OVERRIDE MODE COMMAND; AND
;  286          0416  1 !       'local_mode', WHICH IS THE MODE SET FOR A SINGLE VARIABLE.
;  287          0417  1 !       WHICH MODE IS USED DEPENDS ON THE pointer INTO THE BLOCK
;  288          0418  1 !       dbg$gb_def_mod.
;  289          0419  1 !
;  290          0420  1 !       AT INITIALIZATION TIME, THE FOUR MODE SETTINGS ARE INITIALIZED
;  291          0421  1 !       TO THE SAME values. WHEN THESE values ARE CHANGED BY USER COMMANDS,
;  292          0422  1 !       SOME PROPAGATION OF MODE values MUST MOVE UP AND DOWN THROUGH THE
;  293          0423  1 !       DIFFERENT BLOCKS SO THAT THE ITEMS DISPLAYED OR INTERPRETED WILL
;  294          0424  1 !       HAVE VALID AND TIMELY MODE SETTINGS FOR BOTH RADIX AND LENGTH.
;  295          0425  1 !
;  296          0426  1 !       THE ROUTINES THAT FOLLOW SET THE MODE BLOCK pointerS TO THEIR NEW
;  297          0427  1 !       value AND PROPAGATE values AS NECESSARY.
;  298          0428  1 !
;  299          0429  1 !       THE GENERAL STRATEGY IS AS FOLLOWS:
;  300          0430  1 !       DEFAULT_MODE SETTINGS ARE NEVER CHANGED
;  301          0431  1 !       user_def_mode SETTINGS ARE NEVER CHANGED AS A RESULT OF pointer JIGGLING.
;  302          0432  1 !               THEY ARE ONLY CHANGED EXPLICITLY IN ACTION ROUTINES.
;  303          0433  1 !       override_mode SETTINGS ARE CHANGED TO REFLECT USER-SET DEFAULTS
;  304          0434  1 !               WHEN THEY ARE THE OBJECT OF THE pointer.
;  305          0435  1 !       local_mode SETTINGS ARE COPIED FROM override_mode SETTINGS WHEN THEY
;  306          0436  1 !               ARE THE OBJECT OF THE pointer.
;  307          0437  1 !       WHEN THE pointer IS MOVING UP THE BLOCK, FROM local_mode TOWARD
;  308          0438  1 !               DEFAULT_MODE, NO values ARE PROPAGATED.
;  309          0439  1 !       SETTING OF local_mode AND override_mode TO DEFAULT SETTINGS AT THE END OF
;  310          0440  1 !               A COMMAND LINE IS THE EXPLICIT RESPONSIBILITY OF EOC ACTION
;  311          0441  1 !               ROUTINE.
;  312          0442  1 !--
```

```
;  314          0443  1 GLOBAL ROUTINE dbg$set_mod_lvl (level) : NOVALUE =
;  315          0444  1 !++
;  316          0445  1 ! FUNCTIONAL DESCRIPTION
;  317          0446  1 !       SEE DESCRIPTION ON PAGE 2
;  318          0447  1 !
;  319          0448  1 ! FORMAL PARAMETERS:
;  320          0449  1 !       level   - The structure level to set the pointer to.
;  321          0450  1 !                 user_def_mode
;  322          0451  1 !                 override_mode
;  323          0452  1 !                 local_mode
;  324          0453  1 !
;  325          0454  1 ! IMPLICIT INPUTS:
;  326          0455  1 !       NONE
;  327          0456  1 !
;  328          0457  1 ! IMPLICIT OUTPUTS:
;  329          0458  1 !       NONE
;  330          0459  1 !
;  331          0460  1 ! Routine value:
;  332          0461  1 !       NO value
;  333          0462  1 !
;  334          0463  1 ! SIDE EFFECTS:
;  335          0464  1 !       NONE
;  336          0465  1 !--
;  337          0466  1
;  338          0467  2         BEGIN
;  339          0468  2
;  340          0469  2         LOCAL
;  341          0470  2                 temp_level: REF VECTOR [, BYTE];
;  342          0471  2
;  343          0472  3         IF (dbg$gb_def_mod[.level * mode_lvl_size] LEQA .dbg$gb_mod_ptr)
;  344          0473  3         OR (.level EQL user_def_mode)
;  345          0474  2         THEN
;  346          0475  3             BEGIN
;  347          0476  3             dbg$gb_mod_ptr = dbg$gb_def_mod [.level * mode_lvl_size];
;  348          0477  3             RETURN
;  349          0478  2             END;
;  350          0479  2
;  351          0480  2         SELECT .level OF
;  352          0481  2
;  353          0482  2                 SET
;  354          0483  2
;  355          0484  2                 [override_mode]:
;  356          0485  3                         BEGIN
;  357          0486  3                         temp_level = dbg$gb_def_mod [user_def_mode * mode_lvl_size];
;  358          0487  3                         dbg$gb_mod_ptr = dbg$gb_def_mod [override_mode * mode_lvl_size];
;  359          0488  2                         END;
;  360          0489  2
;  361          0490  2                 [local_mode]:
;  362          0491  3                         BEGIN
;  363          0492  3                         temp_level = .dbg$gb_mod_ptr;
;  364          0493  3                         dbg$gb_mod_ptr = dbg$gb_def_mod [local_mode * mode_lvl_size];
;  365          0494  2                         END;
;  366          0495  2
;  367          0496  2                 TES;
;  368          0497  2         dbg$gb_mod_ptr [mode_radix] = .temp_level [mode_radix];
;  369          0498  2         dbg$gb_mod_ptr [mode_length] = .temp_level [mode_length];
;  370          0499  2         dbg$gb_mod_ptr [mode_symbols] = .temp_level [mode_symbols];
```

```
;  371     0500  2          dbg$gb_mod_ptr [mode_g_floats] = .temp_level [mode_g_floats];
;  372     0501  2          dbg$gb_mod_ptr [mode_instruc] = .temp_level [mode_instruc];
;  373     0502  2          dbg$gb_mod_ptr [mode_ascii] = .temp_level [mode_ascii];
;  374     0503  2          dbg$gb_mod_ptr [mode_scope] = .temp_level [mode_scope];
;  375     0504  2          dbg$gb_mod_ptr [mode_globals] = .temp_level [mode_globals];
;  376     0505  2          dbg$gb_mod_ptr [mode_immediate] = .temp_level [mode_immediate];
;  377     0506  2          dbg$gb_mod_ptr [mode_fortran] = .temp_level [mode_fortran];
;  378     0507  1          END;


                                 000C 00000          .ENTRY  DBG$SET_MOD_LVL, Save R2,R3                             ; 0443
                     53 00000000G   00  9E 00002      MOVAB   DBG$GB_DEF_MOD, R3                                      ;
                     52 00000000G   00  9E 00009      MOVAB   DBG$GB_MOD_PTR, R2                                      ;
                     51        04   AC  D0 00010      MOVL    LEVEL, R1                                               ; 0472
                50   51             0A  C5 00014      MULL3   #10, R1, R0                                             ;
                     50             53  C0 00018      ADDL2   R3, R0                                                  ;
                     62             50  D1 0001B      CMPL    R0, DBG$GB_MOD_PTR                                      ;
                                    05  1B 0001E      BLEQU   1$                                                      ;
                     01             51  D1 00020      CMPL    R1, #1                                                  ; 0473
                                    04  12 00023      BNEQ    2$                                                      ;
                     62             50  D0 00025 1$:  MOVL    R0, DBG$GB_MOD_PTR                                      ; 0476
                                        04 00028      RET                                                             ; 0475
                     02             51  D1 00029 2$:  CMPL    R1, #2                                                  ; 0484
                                    08  12 0002C      BNEQ    3$                                                      ;
                     50        0A   A3  9E 0002E      MOVAB   DBG$GB_DEF_MOD+10, TEMP_LEVEL                           ; 0486
                     62        14   A3  9E 00032      MOVAB   DBG$GB_DEF_MOD+20, DBG$GB_MOD_PTR                       ; 0487
                     03             51  D1 00036 3$:  CMPL    R1, #3                                                  ; 0490
                                    07  12 00039      BNEQ    4$                                                      ;
                     50             62  D0 0003B      MOVL    DBG$GB_MOD_PTR, TEMP_LEVEL                              ; 0492
                     62        1E   A3  9E 0003E      MOVAB   DBG$GB_DEF_MOD+30, DBG$GB_MOD_PTR                       ; 0493
                     51             62  D0 00042 4$:  MOVL    DBG$GB_MOD_PTR, R1                                      ; 0497
                     61             60  D0 00045      MOVL    (TEMP_LEVEL), (R1)                                      ;
               04    A1        04   A0  B0 00048      MOVW    4(TEMP_LEVEL), 4(R1)                                    ; 0502
               06    A1        06   A0  D0 0004D      MOVL    6(TEMP_LEVEL), 6(R1)                                    ; 0504
                                    04 00052      RET                                                             ; 0507

; Routine Size:  83 bytes,    Routine Base:  DBG$CODE + 003F
```

```
;  380          0508  1 GLOBAL ROUTINE dbg$set_mod_def : NOVALUE =
;  381          0509  1 !++
;  382          0510  1 ! FUNCTIONAL DESCRIPTION:
;  383          0511  1 !       CANCELS USER SET DEFAULTS FOR MODE SETTINGS.
;  384          0512  1 !       RESETS ALL MODE LEVELS TO LANGUAGE DEFAULTS.
;  385          0513  1 !       Called from  DBG$INIT_DEBUG as part of the
;  386          0514  1 !       once-only initialization phase of DEBUG.
;  387          0515  1 !
;  388          0516  1 ! FORMAL PARAMETERS:
;  389          0517  1 !       NONE
;  390          0518  1 !
;  391          0519  1 ! IMPLICIT INPUTS:
;  392          0520  1 !       THE DEFAULT MODES
;  393          0521  1 !
;  394          0522  1 ! IMPLICIT OUTPUTS:
;  395          0523  1 !       NONE
;  396          0524  1 !
;  397          0525  1 ! Routine value:
;  398          0526  1 !       NOVALUE
;  399          0527  1 !
;  400          0528  1 ! SIDE EFFECTS:
;  401          0529  1 !       NONE
;  402          0530  1 !--
;  403          0531  1
;  404          0532  2         BEGIN
;  405          0533  2
;  406          0534  2         !+
;  407          0535  2         ! Set up the MODE data structure
;  408          0536  2         !-
;  409          0537  2
;  410          0538  2         dbg$gb_mod_ptr = dbg$gb_def_mod [default_mode * mode_lvl_size];
;  411          0539  2
;  412          0540  2         dbg$gb_mod_ptr [mode_radix] = dbg$k_default;
;  413          0541  2         dbg$gb_mod_ptr [mode_length] = long_length;
;  414          0542  2         dbg$gb_mod_ptr [mode_symbols] = TRUE;
;  415          0543  2         dbg$gb_mod_ptr [mode_g_floats] = FALSE;
;  416          0544  2         dbg$gb_mod_ptr [mode_instruc] = FALSE;
;  417          0545  2         dbg$gb_mod_ptr [mode_ascii] = FALSE;
;  418          0546  2         dbg$gb_mod_ptr [mode_scope] = TRUE;
;  419          0547  2         dbg$gb_mod_ptr [mode_globals] = FALSE;
;  420          0548  2         dbg$gb_mod_ptr [mode_immediate] = TRUE;
;  421          0549  2         dbg$gb_mod_ptr [mode_fortran] = 0;
;  422          0550  2         dbg$init_modes (user_def_mode, default_mode);
;  423          0551  1         END;


                                 0004 00000         .ENTRY  DBG$SET_MOD_DEF, Save R2                        ; 0508
        52 00000000G   00  9E 00002         MOVAB   DBG$GB_MOD_PTR, R2                              ;
        62 00000000G   00  9E 00009         MOVAB   DBG$GB_DEF_MOD, DBG$GB_MOD_PTR                  ; 0538
        50             62  D0 00010         MOVL    DBG$GB_MOD_PTR, R0                              ; 0540
        60 00010401    8F  D0 00013         MOVL    #66561, (R0)                                    ;
  04    A0      0100   8F  B0 0001A         MOVW    #256, 4(R0)                                     ; 0545
  06    A0 00010000    8F  D0 00020         MOVL    #65536, 6(R0)                                   ; 0547
        7E             01  7D 00028         MOVQ    #1, -(SP)                                       ; 0550
```

```
                FF3E    CF              02  FB 0002B           CALLS   #2, DBG$INIT_MODES                ;
                                            04 00030           RET                                      ; 0551

; Routine Size:  49 bytes,    Routine Base:  DBG$CODE + 0092
```

```
;  425          0552  1 GLOBAL ROUTINE dbg$show_mode : NOVALUE =
;  426          0553  1 !++
;  427          0554  1 ! FUNCTIONAL DESCRIPTION:
;  428          0555  1 !       ACTION ROUTINE TO SHOW MODES
;  429          0556  1 !
;  430          0557  1 ! FORMAL PARAMETERS:
;  431          0558  1 !       NONE
;  432          0559  1 !
;  433          0560  1 ! IMPLICIT INPUTS:
;  434          0561  1 !       THE POINTER TO THE CURRENT MODE STRUCTURE.
;  435          0562  1 !
;  436          0563  1 ! IMPLICIT OUTPUTS:
;  437          0564  1 !       NONE
;  438          0565  1 !
;  439          0566  1 ! Routine value:
;  440          0567  1 !       NOVALUE
;  441          0568  1 !
;  442          0569  1 ! SIDE EFFECTS:
;  443          0570  1 !       THE MODES ARE DISPLAYED ON THE OUTPUT DEVICE
;  444          0571  1 !--
;  445          0572  2     BEGIN
;  446          0573  2     dbg$print(UPLIT BYTE(%ASCIC 'modes: '));
;  447          0574  2     IF NOT .dbg$gb_mod_ptr[mode_symbols]
;  448          0575  2     THEN
;  449          0576  2         dbg$print(UPLIT BYTE(%ASCIC 'no'));
;  450          0577  2     dbg$print(UPLIT BYTE(%ASCIC 'symbolic, '));
;  451          0578  2     IF NOT .dbg$gb_mod_ptr[mode_g_floats]
;  452          0579  2     THEN
;  453          0580  2         dbg$print(UPLIT BYTE(%ASCIC 'd_float, '))
;  454          0581  2     ELSE
;  455          0582  2         dbg$print(UPLIT BYTE(%ASCIC 'g_float, '));
;  456          0583  2     IF NOT .dbg$gl_screen_mode
;  457          0584  2     THEN
;  458          0585  2         dbg$print(UPLIT BYTE(%ASCIC 'no'));
;  459          0586  2     dbg$print(UPLIT BYTE(%ASCIC 'screen, '));
;  460          0587  2     IF NOT .dbg$gb_keypad_input
;  461          0588  2     THEN
;  462          0589  2         dbg$print(UPLIT BYTE(%ASCIC 'no'));
;  463          0590  2     dbg$print(UPLIT BYTE(%ASCIC 'keypad'));
;  464          0591  2     dbg$newline();
;  465          0592  2     dbg$show_radix(0);
;  466          0593  1     END;


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

                     20  3A  73  65  64  6F  6D  07  00009 P.AAB:  .ASCII  <7>\modes: \                ;
                                 6F  6E  02  00011 P.AAC:  .ASCII  <2>\no\                       ;
20   2C  63  69  6C  6F  62  6D  79  73  0A  00014 P.AAD:  .ASCII  <10>\symbolic, \                 ;
     20  2C  74  61  6F  6C  66  5F  64  09  0001F P.AAE:  .ASCII  <9>\d_float, \                  ;
     20  2C  74  61  6F  6C  66  5F  67  09  00029 P.AAF:  .ASCII  <9>\g_float, \                  ;
                                 6F  6E  02  00033 P.AAG:  .ASCII  <2>\no\                       ;
         20  2C  6E  65  65  72  63  73  08  00036 P.AAH:  .ASCII  <8>\screen, \                  ;
                                 6F  6E  02  0003F P.AAI:  .ASCII  <2>\no\                       ;
                 64  61  70  79  65  6B  06  00042 P.AAJ:  .ASCII  <6>\keypad\                   ;
```

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

                                          001C 00000    .ENTRY  DBG$SHOW_MODE, Save R2,R3,R4                    ; 0552
                54 00000000G   00   9E  00002          MOVAB   DBG$GB_MOD_PTR, R4                              ;
                53 00000000G   00   9E  00009          MOVAB   DBG$PRINT, R3                                   ;
                52 00000000'   EF   9E  00010          MOVAB   P.AAB, R2                                       ;
                               52   DD  00017          PUSHL   R2                                              ; 0573
                63             01   FB  00019          CALLS   #1, DBG$PRINT                                   ;
                50             64   D0  0001C          MOVL    DBG$GB_MOD_PTR, R0                              ; 0574
                06       02    A0   E8  0001F          BLBS    2(R0), 1$                                       ;
                         08    A2   9F  00023          PUSHAB  P.AAC                                           ; 0576
                63             01   FB  00026          CALLS   #1, DBG$PRINT                                   ;
                         0B    A2   9F  00029 1$:      PUSHAB  P.AAD                                           ; 0577
                63             01   FB  0002C          CALLS   #1, DBG$PRINT                                   ;
                50             64   D0  0002F          MOVL    DBG$GB_MOD_PTR, R0                              ; 0578
                05       09    A0   E8  00032          BLBS    9(R0), 2$                                       ;
                         16    A2   9F  00036          PUSHAB  P.AAE                                           ; 0580
                               03   11  00039          BRB     3$                                              ;
                         20    A2   9F  0003B 2$:      PUSHAB  P.AAF                                           ; 0582
                63             01   FB  0003E 3$:      CALLS   #1, DBG$PRINT                                   ;
                06 00000000G   00   E8  00041          BLBS    DBG$GL_SCREEN_MODE, 4$                          ; 0583
                         2A    A2   9F  00048          PUSHAB  P.AAG                                           ; 0585
                63             01   FB  0004B          CALLS   #1, DBG$PRINT                                   ;
                         2D    A2   9F  0004E 4$:      PUSHAB  P.AAH                                           ; 0586
                63             01   FB  00051          CALLS   #1, DBG$PRINT                                   ;
                06 00000000G   00   E8  00054          BLBS    DBG$GB_KEYPAD_INPUT, 5$                         ; 0587
                         36    A2   9F  0005B          PUSHAB  P.AAI                                           ; 0589
                63             01   FB  0005E          CALLS   #1, DBG$PRINT                                   ;
                         39    A2   9F  00061 5$:      PUSHAB  P.AAJ                                           ; 0590
                63             01   FB  00064          CALLS   #1, DBG$PRINT                                   ;
          00000000G   00       00   FB  00067          CALLS   #0, DBG$NEWLINE                                 ; 0591
                               7E   D4  0006E          CLRL    -(SP)                                           ; 0592
          00000000G   00       01   FB  00070          CALLS   #1, DBG$SHOW_RADIX                              ;
                               04   00077              RET                                                     ; 0593

; Routine Size:  120 bytes,    Routine Base:  DBG$CODE + 00C3
```

```
;  468          0594  1 GLOBAL ROUTINE DBG$INIT_STEP (GOAL_LEVEL, SOURCE_LEVEL) : NOVALUE =
;  469          0595  1 !++
;  470          0596  1 ! FUNCTIONAL DESCRIPTION:
;  471          0597  1 !       Sets all levels from override level to the goal level specified
;  472          0598  1 !       with the step settings of the source level.
;  473          0599  1 !
;  474          0600  1 !
;  475          0601  1 ! INPUTS:
;  476          0602  1 !       goal_level      - highest level to set
;  477          0603  1 !       source_level    - level from which to obtain step settings
;  478          0604  1 !
;  479          0605  1 ! IMPLICIT INPUTS:
;  480          0606  1 !       none
;  481          0607  1 !
;  482          0608  1 ! OUTPUTS:
;  483          0609  1 !       NONE
;  484          0610  1 !
;  485          0611  1 ! IMPLICIT OUTPUTS:
;  486          0612  1 !       THE step settings of the override level to the goal level
;  487          0613  1 !       are reset to the step settings of the source level.
;  488          0614  1 !
;  489          0615  1 ! Routine value:
;  490          0616  1 !       NOVALUE
;  491          0617  1 !
;  492          0618  1 ! SIDE EFFECTS:
;  493          0619  1 !       NONE
;  494          0620  1 !--
;  495          0621  2     BEGIN
;  496          0622  2     LOCAL
;  497          0623  2         TEMP_LEVEL :    REF EVENT$STEPPING_DESCRIPTOR;
;  498          0624  2
;  499          0625  2     TEMP_LEVEL = DBG$GB_DEF_STP [.SOURCE_LEVEL,
;  500          0626  2                                  EVENT$A_STEPPING_DESCRIPTOR
;  501          0627  2                                 ];
;  502          0628  2     DECR I FROM OVERRIDE_STEP TO .GOAL_LEVEL DO
;  503          0629  3         BEGIN
;  504          0630  3         DBG$GB_STP_PTR = DBG$GB_DEF_STP [.I, EVENT$A_STEPPING_DESCRIPTOR];
;  505          0631  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] =
;  506          0632  3             .TEMP_LEVEL [EVENT$V_STEPPING_LINE];
;  507          0633  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] =
;  508          0634  3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSYSTEM];
;  509          0635  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] =
;  510          0636  3             .TEMP_LEVEL [EVENT$V_STEPPING_OVER];
;  511          0637  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] =
;  512          0638  3             .TEMP_LEVEL [EVENT$V_STEPPING_SOURCE];
;  513          0639  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] =
;  514          0640  3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSILENT];
;  515          0641  3
;  516          0642  3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] =
;  517          0643  3             .TEMP_LEVEL [EVENT$B_STEPPING_KIND];
;  518          0644  3         DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] =
;  519          0645  3             .TEMP_LEVEL [EVENT$L_STEPPING_OP_LIST];
;  520          0646  2         END;
;  521          0647  1     END;
```

```
                                                003C 00000          .ENTRY  DBG$INIT_STEP, Save R2,R3,R4,R5                 ; 0594
                                     55 00000000G   00   9E 00002   MOVAB   DBG$GB_DEF_STP, R5                            ;
                                     54 00000000G   00   9E 00009   MOVAB   DBG$GB_STP_PTR, R4                           ;
                                     50         08   AC   D0 00010   MOVL    SOURCE_LEVEL, R0                          ; 0625
                                     51           6540   7E 00014   MOVAQ   DBG$GB_DEF_STP[R0], TEMP_LEVEL            ; 0627
                                     52             02   D0 00018   MOVL    #2, I                                     ; 0645
                                                    40   11 0001B   BRB     2$                                        ;
                                     64           6542   7E 0001D 1$:  MOVAQ   DBG$GB_DEF_STP[I], DBG$GB_STP_PTR        ; 0630
                                     50             64   D0 00021   MOVL    DBG$GB_STP_PTR, R0                        ; 0631
      01    A0          01           00         01   A1   F0 00024   INSV    1(TEMP_LEVEL), #0, #1, 1(R0)              ; 0632
            53          61           01           0B   EF 0002B   EXTZV   #11, #1, (TEMP_LEVEL), R3                    ; 0634
            60          01           0B             53   F0 00030   INSV    R3, #11, #1, (R0)                         ;
            53          61           01             09   EF 00035   EXTZV   #9, #1, (TEMP_LEVEL), R3                   ; 0636
            60          01           09             53   F0 0003A   INSV    R3, #9, #1, (R0)                          ;
            53          61           01             0A   EF 0003F   EXTZV   #10, #1, (TEMP_LEVEL), R3                  ; 0638
            60          01           0A             53   F0 00044   INSV    R3, #10, #1, (R0)                         ;
            53          61           01             0C   EF 00049   EXTZV   #12, #1, (TEMP_LEVEL), R3                  ; 0640
            60          01           0C             53   F0 0004E   INSV    R3, #12, #1, (R0)                         ;
                                     60             61   90 00053   MOVB    (TEMP_LEVEL), (R0)                        ; 0643
                              04     A0         04   A1   D0 00056   MOVL    4(TEMP_LEVEL), 4(R0)                      ; 0645
                                                    52   D7 0005B   DECL    I                                         ; 0628
                              04     AC             52   D1 0005D 2$:  CMPL    I, GOAL_LEVEL                             ;
                                                    BA   18 00061   BGEQ    1$                                        ;
                                                         04 00063   RET                                               ; 0647

; Routine Size:  100 bytes,    Routine Base:  DBG$CODE + 013B
```

```
;   523          0648  1 GLOBAL ROUTINE DBG$SET_STP_DEF : NOVALUE =
;   524          0649  1 !++
;   525          0650  1 ! FUNCTIONAL DESCRIPTION:
;   526          0651  1 !       CANCELS USER SET DEFAULTS FOR STEP SETTINGS.
;   527          0652  1 !       RESETS ALL STEP LEVELS TO SYSTEM DEFAULTS FOR THE CURRENT LANGUAGE.
;   528          0653  1 !
;   529          0654  1 !
;   530          0655  1 ! INPUTS:
;   531          0656  1 !       NONE
;   532          0657  1 !
;   533          0658  1 ! IMPLICIT INPUTS:
;   534          0659  1 !       THE DEFAULT STEP SETTINGS
;   535          0660  1 !
;   536          0661  1 ! OUTPUTS:
;   537          0662  1 !       NONE
;   538          0663  1 !
;   539          0664  1 ! IMPLICIT OUTPUTS:
;   540          0665  1 !       NONE
;   541          0666  1 !
;   542          0667  1 ! Routine value:
;   543          0668  1 !       NOVALUE
;   544          0669  1 !
;   545          0670  1 ! SIDE EFFECTS:
;   546          0671  1 !       RESETS DEFAULT values FOR STEP.
;   547          0672  1 !--
;   548          0673  2     BEGIN
;   549          0674  2
;   550          0675  2
;   551          0676  2     ! Set up the step data structure
;   552          0677  2     !
;   553          0678  2     DBG$GB_STP_PTR = DBG$GB_DEF_STP [DEFAULT_STEP,
;   554          0679  2                                      EVENT$A_STEPPING_DESCRIPTOR
;   555          0680  2                                     ];
;   556          0681  2     IF .DBG$GB_LANGUAGE EQL DBG$K_MACRO
;   557          0682  2     THEN
;   558          0683  3         BEGIN
;   559          0684  3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] = EVENT$K_INS_EVRY;
;   560          0685  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] = FALSE;
;   561          0686  3         END
;   562          0687  2     ELSE
;   563          0688  3         BEGIN
;   564          0689  3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] = EVENT$K_INS_LINE;
;   565          0690  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] = TRUE;
;   566          0691  2         END;
;   567          0692  2     DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] = TRUE;
;   568          0693  2     DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] = TRUE;
;   569          0694  2     DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] = TRUE;
;   570          0695  2     DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] = TRUE;
;   571          0696  2
;   572          0697  2     DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] = 0;
;   573          0698  2
;   574          0699  2     DBG$INIT_STEP (USER_DEF_STEP, DEFAULT_STEP);
;   575          0700  1     END;
```

```
                                    0004 00000          .ENTRY  DBG$SET_STP_DEF, Save R2                        ; 0648
                52 00000000G    00   9E 00002          MOVAB   DBG$GB_STP_PTR, R2                              ;
                62 00000000G    00   9E 00009          MOVAB   DBG$GB_DEF_STP, DBG$GB_STP_PTR                  ; 0680
                50                   62 D0 00010      MOVL    DBG$GB_STP_PTR, R0                              ; 0684
                   00000000G    00   95 00013          TSTB    DBG$GB_LANGUAGE                                 ; 0681
                                     09 12 00019        BNEQ    1$                                              ;
                60                   08 90 0001B        MOVB    #8, (R0)                                        ; 0684
             01 A0                   01 8A 0001E        BICB2   #1, 1(R0)                                       ; 0685
                                     07 11 00022        BRB     2$                                              ; 0681
                60                   07 90 00024 1$:    MOVB    #7, (R0)                                        ; 0689
             01 A0                   01 88 00027        BISB2   #1, 1(R0)                                       ; 0690
             01 A0                   1E 88 0002B 2$:    BISB2   #30, 1(R0)                                      ; 0695
                      04         A0 D4 0002F            CLRL    4(R0)                                           ; 0697
                7E                   01 7D 00032        MOVQ    #1, -(SP)                                       ; 0699
          FF62  CF                   02 FB 00035        CALLS   #2, DBG$INIT_STEP                               ;
                                     04 0003A           RET                                                     ; 0700

; Routine Size:  59 bytes,    Routine Base:  DBG$CODE + 019F
```

```
;  577     0701  1 GLOBAL ROUTINE DBG$SET_STP_LVL (LEVEL) : NOVALUE =
;  578     0702  1 !++
;  579     0703  1 ! FUNCTIONAL DESCRIPTION
;  580     0704  1 !       This routine sets the pointer to the step structure to the specified
;  581     0705  1 !       level.  If the level is not user_def_step the step values there are
;  582     0706  1 !       copied to override_step level.  If the level is user_def_step the
;  583     0707  1 !       pointer is just reset to that level.
;  584     0708  1 !
;  585     0709  1 ! FORMAL PARAMETERS:
;  586     0710  1 !       LEVEL   - The new step structure level.
;  587     0711  1 !                 user_def_step
;  588     0712  1 !                 override_step
;  589     0713  1 !
;  590     0714  1 ! IMPLICIT INPUTS:
;  591     0715  1 !       NONE
;  592     0716  1 !
;  593     0717  1 ! IMPLICIT OUTPUTS:
;  594     0718  1 !       NONE
;  595     0719  1 !
;  596     0720  1 ! Routine value:
;  597     0721  1 !       NO value
;  598     0722  1 !
;  599     0723  1 ! SIDE EFFECTS:
;  600     0724  1 !       NONE
;  601     0725  1 !--
;  602     0726  2     BEGIN
;  603     0727  2
;  604     0728  2     LOCAL
;  605     0729  2         TEMP_LEVEL: REF EVENT$STEPPING_DESCRIPTOR;
;  606     0730  2
;  607     0731  2     IF .LEVEL EQL USER_DEF_STEP
;  608     0732  2     THEN
;  609     0733  2         DBG$GB_STP_PTR = DBG$GB_DEF_STP [USER_DEF_STEP,
;  610     0734  2                                          EVENT$A_STEPPING_DESCRIPTOR
;  611     0735  2                                         ]
;  612     0736  2     ELSE
;  613     0737  3         BEGIN
;  614     0738  3         TEMP_LEVEL = DBG$GB_DEF_STP [USER_DEF_STEP,
;  615     0739  3                                      EVENT$A_STEPPING_DESCRIPTOR
;  616     0740  3                                     ];
;  617     0741  3         DBG$GB_STP_PTR = DBG$GB_DEF_STP [OVERRIDE_STEP,
;  618     0742  3                                          EVENT$A_STEPPING_DESCRIPTOR
;  619     0743  3                                         ];
;  620     0744  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] =
;  621     0745  3             .TEMP_LEVEL [EVENT$V_STEPPING_LINE];
;  622     0746  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] =
;  623     0747  3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSYSTEM];
;  624     0748  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] =
;  625     0749  3             .TEMP_LEVEL [EVENT$V_STEPPING_OVER];
;  626     0750  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] =
;  627     0751  3             .TEMP_LEVEL [EVENT$V_STEPPING_SOURCE];
;  628     0752  3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] =
;  629     0753  3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSILENT];
;  630     0754  3
;  631     0755  3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] =
;  632     0756  3             .TEMP_LEVEL [EVENT$B_STEPPING_KIND];
;  633     0757  3         DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] =
```

```
;  634          0758  3              .TEMP_LEVEL [EVENT$L_STEPPING_OP_LIST];
;  635          0759  2          END;
;  636          0760  1      END;


                                                001C 00000          .ENTRY  DBG$SET_STP_LVL, Save R2,R3,R4                  ; 0701
                                    54 00000000G   00    9E 00002   MOVAB   DBG$GB_DEF_STP+8, R4                          ;
                                    53 00000000G   00    9E 00009   MOVAB   DBG$GB_STP_PTR, R3                           ;
                                    01         04    AC   D1 00010   CMPL    LEVEL, #1                                   ; 0731
                                                     04   12 00014   BNEQ    1$                                          ;
                                    63                   64   9E 00016   MOVAB   DBG$GB_DEF_STP+8, DBG$GB_STP_PTR       ; 0735
                                                         04   00019   RET                                                 ; 0733
                                    51                   64   9E 0001A 1$:    MOVAB   DBG$GB_DEF_STP+8, TEMP_LEVEL       ; 0740
                                    63             08    A4   9E 0001D   MOVAB   DBG$GB_DEF_STP+16, DBG$GB_STP_PTR      ; 0743
                                    50                   63   D0 00021   MOVL    DBG$GB_STP_PTR, R0                      ; 0744
      01    A0          01                   00         01    A1   F0 00024   INSV    1(TEMP_LEVEL), #0, #1, 1(R0)        ; 0745
            52          61                   01               0B   EF 0002B   EXTZV   #11, #1, (TEMP_LEVEL), R2           ; 0747
            60          01                   0B               52   F0 00030   INSV    R2, #11, #1, (R0)                   ;
            52          61                   01               09   EF 00035   EXTZV   #9, #1, (TEMP_LEVEL), R2           ; 0749
            60          01                   09               52   F0 0003A   INSV    R2, #9, #1, (R0)                    ;
            52          61                   01               0A   EF 0003F   EXTZV   #10, #1, (TEMP_LEVEL), R2          ; 0751
            60          01                   0A               52   F0 00044   INSV    R2, #10, #1, (R0)                   ;
            52          61                   01               0C   EF 00049   EXTZV   #12, #1, (TEMP_LEVEL), R2          ; 0753
            60          01                   0C               52   F0 0004E   INSV    R2, #12, #1, (R0)                   ;
                                             60               61   90 00053   MOVB    (TEMP_LEVEL), (R0)                  ; 0756
                                    04   A0        04         A1   D0 00056   MOVL    4(TEMP_LEVEL), 4(R0)                ; 0758
                                                               04   0005B   RET                                         ; 0760

; Routine Size:  92 bytes,    Routine Base:  DBG$CODE + 01DA
```

```
;  638          0761  1 GLOBAL ROUTINE DBG$SHOW_STEP : NOVALUE =
;  639          0762  1 !++
;  640          0763  1 ! FUNCTIONAL DESCRIPTION:
;  641          0764  1 !       ACTION ROUTINE TO DISPLAY THE CURRENT SETTING OF THE
;  642          0765  1 !       "STEP TYPE" DATA STRUCTURE
;  643          0766  1 !
;  644          0767  1 ! INPUTS:
;  645          0768  1 !       NONE
;  646          0769  1 !
;  647          0770  1 ! IMPLICIT INPUTS·
;  648          0771  1 !       none.
;  649          0772  1 !
;  650          0773  1 ! OUTPUTS:
;  651          0774  1 !       NONE
;  652          0775  1 !
;  653          0776  1 ! IMPLICIT OUTPUTS:
;  654          0777  1 !       NONE
;  655          0778  1 !
;  656          0779  1 ! Routine value:
;  657          0780  1 !       NOVALUE
;  658          0781  1 !
;  659          0782  1 ! SIDE EFFECTS:
;  660          0783  1 !       THE STEP SETTINGS ARE DISPLAYED ON THE OUTPUT DEVICE
;  661          0784  1 !--
;  662          0785  2     BEGIN
;  663          0786  2
;  664          0787  2
;  665          0788  2     ! Setup a UPLIT %ASCIC string.
;  666          0789  2     !
;  667          0790  2     MACRO
;  668        M 0791  2         $AC (STRING) =
;  669        M 0792  2             UPLIT BYTE (%ASCIC STRING)
;  670          0793  2         %;
;  671          0794  2
;  672          0795  2
;  673          0796  2     ! Step type...
;  674          0797  2     !
;  675          0798  2     DBG$PRINT   ($AC    ('step type: '));
;  676          0799  2
;  677          0800  2
;  678          0801  2     ! [NO] SYSTEM....
;  679          0802  2     !
;  680          0803  2     IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM]
;  681          0804  2     THEN
;  682          0805  2         DBG$PRINT   ($AC    ('no'));
;  683          0806  2     DBG$PRINT   ($AC    ('system, '));
;  684          0807  2
;  685          0808  2
;  686          0809  2     ! [NO] SOURCE....
;  687          0810  2     !
;  688          0811  2     IF NOT .DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE]
;  689          0812  2     THEN
;  690          0813  2         DBG$PRINT   ($AC    ('no'));
;  691          0814  2     DBG$PRINT   ($AC    ('source, '));
;  692          0815  2
;  693          0816  2
;  694          0817  2     ! [NO] SILENT....
```

```
;  695     0818  2      !
;  696     0819  2      IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT]
;  697     0820  2      THEN
;  698     0821  2          DBG$PRINT   ($AC    ('no'));
;  699     0822  2      DBG$PRINT   ($AC    ('silent, '));
;  700     0823  2
;  701     0824  2
;  702     0825  2      ! INTO/OVER....
;  703     0826  2      !
;  704     0827  2      IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER]
;  705     0828  2      THEN
;  706     0829  2          DBG$PRINT   ($AC    ('over'))
;  707     0830  2      ELSE
;  708     0831  2          DBG$PRINT   ($AC    ('into'));
;  709     0832  2      DBG$PRINT   ($AC    (' routine calls, '));
;  710     0833  2
;  711     0834  2
;  712     0835  2      ! Display the instruction type (calls, branches, all, or user
;  713     0836  2      ! list.
;  714     0837  2      !
;  715     0838  2      DBG$PRINT   ($AC ('by !AC'),
;  716     0839  2                   SELECTONE .DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] OF
;  717     0840  2                      SET
;  718     0841  2                      [EVENT$K_INS_CALL] :
;  719     0842  2                          $AC ('calls:');
;  720     0843  2                      [EVENT$K_INS_BRAN] :
;  721     0844  2                          $AC ('branches:');
;  722     0845  2                      [EVENT$K_EXC_EXC] :
;  723     0846  2                          $AC ('exceptions');
;  724     0847  2                      [EVENT$K_INS_EVRY] :
;  725     0848  2                          $AC ('instruction');
;  726     0849  2                      [EVENT$K_INS_USER] :
;  727     0850  2                          $AC ('instruction(s):');
;  728     0851  2                      [EVENT$K_INS_LINE] :
;  729     0852  2                          $AC ('line');
;  730     0853  2                      [EVENT$K_ACC_RTRN] :
;  731     0854  2                          $AC ('return');
;  732     0855  2                      TES
;  733     0856  2                  );
;  734     0857  2
;  735     0858  2
;  736     0859  2      ! If this entry is an instruction type (/CALL, /BRANCH, etc.) and
;  737     0860  2      ! has an opcode bit map (i.e., is not for all instructions),
;  738     0861  2      ! display the opcode mnemonics indicated.
;  739     0862  2      !
;  740     0863  2      IF .DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] NEQA 0
;  741     0864  2      THEN
;  742     0865  3          BEGIN
;  743     0866  3          LOCAL
;  744     0867  3              OPCODE_LIST :   REF BITVECTOR [512],
;  745     0868  3              COLUMN;
;  746     0869  3
;  747     0870  3          ! Point to the opcode list, and set the column to 0.
;  748     0871  3          !
;  749     0872  3          OPCODE_LIST = .DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST];
;  750     0873  3          COLUMN = -1;
;  751     0874  3          INCR OPCODE FROM 0 TO 511 DO
```

```
;  752          0875  3            IF .OPCODE_LIST [.OPCODE] THEN
;  753          0876  4              BEGIN
;  754          0877  4              LOCAL INDEX;
;  755          0878  4
;  756          0879  4              IF (COLUMN = (.COLUMN + 1) AND 7) EQLU 0 THEN DBG$NEWLINE ();
;  757          0880  4              INDEX = .DBG$OPCODE_KIND_TABLE[.OPCODE];
;  758          0881  4              DBG$PRINT ($AC ('!_!AD'),6,DBG$OPCODE_NAME_TABLE[.INDEX,0,0,0,0]);
;  759          0882  3              END;
;  760          0883  3          DBG$NEWLINE ();
;  761          0884  3          END
;  762          0885  3
;  763          0886  3
;  764          0887  3      ! Otherwise, just print the line.
;  765          0888  3      !
;  766          0889  2      ELSE
;  767          0890  2          DBG$NEWLINE ();
;  768          0891  1      END;


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

          20 3A 65 70 79 74 20 70 65 74 73 0B  00049 P.AAK:   .ASCII  <11>\step type: \                    ;
                                        6F 6E 02  00055 P.AAL:   .ASCII  <2>\no\                           ;
                20 2C 6D 65 74 73 79 73 08  00058 P.AAM:   .ASCII  <8>\system, \                     ;
                                        6F 6E 02  00061 P.AAN:   .ASCII  <2>\no\                           ;
                20 2C 65 63 72 75 6F 73 08  00064 P.AAO:   .ASCII  <8>\source, \                     ;
                                        6F 6E 02  0006D P.AAP:   .ASCII  <2>\no\                           ;
                20 2C 74 6E 65 6C 69 73 08  00070 P.AAQ:   .ASCII  <8>\silent, \                     ;
                                     72 65 76 6F 04  00079 P.AAR:   .ASCII  <4>\over\                      ;
                                     6F 74 6E 69 04  0007E P.AAS:   .ASCII  <4>\into\                      ;
73 6C 6C 61 63 20 65 6E 69 74 75 6F 72 20 10  00083 P.AAT:   .ASCII  <16>\ routine calls, \               ;
                                              20 2C  00092
                                  43 41 21 20 79 62 06  00094 P.AAU:   .ASCII  <6>\by !AC\                ;
                                  3A 73 6C 6C 61 63 06  0009B P.AAV:   .ASCII  <6>\calls:\                ;
                              3A 73 65 68 63 6E 61 72 62 09  000A2 P.AAW:   .ASCII  <9>\branches:\         ;
                           73 6E 6F 69 74 70 65 63 78 65 0A  000AC P.AAX:   .ASCII  <10>\exceptions\       ;
                        6E 6F 69 74 63 75 72 74 73 6E 69 0B  000B7 P.AAY:   .ASCII  <11>\instruction\      ;
29 73 28 6E 6F 69 74 63 75 72 74 73 6E 69 0F  000C3 P.AAZ:   .ASCII  <15>\instruction(s):\              ;
                                              3A  000D2
                                  65 6E 69 6C 04  000D3 P.ABA:   .ASCII  <4>\line\                      ;
                                  6E 72 75 74 65 72 06  000D8 P.ABB:   .ASCII  <6>\return\               ;
                                     44 41 21 5F 21 05  000DF P.ABC:   .ASCII  <5>\!_!AD\                ;


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

                                                 01FC 00000         .ENTRY  DBG$SHOW_STEP, Save R2,R3,R4,R5,R6,R7,R8    ; 0761
                                  58 00000000G   00  9E 00002       MOVAB   DBG$NEWLINE, R8                            ;
                                  57 00000000G   00  9E 00009       MOVAB   DBG$GB_STP_PTR, R7                         ;
                                  56 00000000G   00  9E 00010       MOVAB   DBG$PRINT, R6                              ;
                                  55 00000000'   EF  9E 00017       MOVAB   P.AAK, R5                                  ;
                                                 55  DD 0001E       PUSHL   R5                                         ; 0798
                                  66             01  FB 00020       CALLS   #1, DBG$PRINT                              ;
                                  50             67  D0 00023       MOVL    DBG$GB_STP_PTR, R0                         ; 0803
                        06        60             0B  E1 00026       BBC     #11, (R0), 1$                              ;
```

```
                0C   A5  9F 0002A          PUSHAB  P.AAL                                   ; 0805
            66      01  FB 0002D          CALLS   #1, DBG$PRINT
                0F   A5  9F 00030 1$:      PUSHAB  P.AAM                                   ; 0806
            66      01  FB 00033          CALLS   #1, DBG$PRINT
            50      67  D0 00036          MOVL    DBG$GB_STP_PTR, R0                      ; 0811
    06      60      0A  E0 00039          BBS     #10, (R0), 2$
                18   A5  9F 0003D          PUSHAB  P.AAN                                   ; 0813
            66      01  FB 00040          CALLS   #1, DBG$PRINT
                1B   A5  9F 00043 2$:      PUSHAB  P.AAO                                   ; 0814
            66      01  FB 00046          CALLS   #1, DBG$PRINT
            50      67  D0 00049          MOVL    DBG$GB_STP_PTR, R0                      ; 0819
    06      60      0C  E1 0004C          BBC     #12, (R0), 3$
                24   A5  9F 00050          PUSHAB  P.AAP                                   ; 0821
            66      01  FB 00053          CALLS   #1, DBG$PRINT
                27   A5  9F 00056 3$:      PUSHAB  P.AAQ                                   ; 0822
            66      01  FB 00059          CALLS   #1, DBG$PRINT
            50      67  D0 0005C          MOVL    DBG$GB_STP_PTR, R0                      ; 0827
    05      60      09  E1 0005F          BBC     #9, (R0), 4$
                30   A5  9F 00063          PUSHAB  P.AAR                                   ; 0829
                    03  11 00066          BRB     5$
                35   A5  9F 00068 4$:      PUSHAB  P.AAS                                   ; 0831
            66      01  FB 0006B 5$:      CALLS   #1, DBG$PRINT
                3A   A5  9F 0006E          PUSHAB  P.AAT                                   ; 0832
            66      01  FB 00071          CALLS   #1, DBG$PRINT
            50      67  D0 00074          MOVL    DBG$GB_STP_PTR, R0                      ; 0839
            05      60  91 00077          CMPB    (R0), #5                                ; 0841
                    06  12 0007A          BNEQ    6$
            51  52   A5  9E 0007C          MOVAB   P.AAV, R1                               ; 0842
                    36  11 00080          BRB     11$
            06      60  91 00082 6$:      CMPB    (R0), #6                                ; 0843
                    06  12 00085          BNEQ    7$
            51  59   A5  9E 00087          MOVAB   P.AAW, R1                               ; 0844
                    2B  11 0008B          BRB     11$
            0F      60  91 0008D 7$:      CMPB    (R0), #15                               ; 0845
                    06  12 00090          BNEQ    8$
            51  63   A5  9E 00092          MOVAB   P.AAX, R1                               ; 0846
                    20  11 00096          BRB     11$
            08      60  91 00098 8$:      CMPB    (R0), #8                                ; 0847
                    06  12 0009B          BNEQ    9$
            51  6E   A5  9E 0009D          MOVAB   P.AAY, R1                               ; 0848
                    15  11 000A1          BRB     11$
            09      60  91 000A3 9$:      CMPB    (R0), #9                                ; 0849
                    06  12 000A6          BNEQ    10$
            51  7A   A5  9E 000A8          MOVAB   P.AAZ, R1                               ; 0850
                    0A  11 000AC          BRB     11$
            07      60  91 000AE 10$:     CMPB    (R0), #7                                ; 0851
                    09  12 000B1          BNEQ    12$
            51  008A  C5  9E 000B3          MOVAB   P.ABA, R1                               ; 0852
                    51  DD 000B8 11$:      PUSHL   R1
                    11  11 000BA          BRB     14$
            04      60  91 000BC 12$:     CMPB    (R0), #4                                ; 0853
                    05  13 000BF          BEQL    13$
            7E      01  CE 000C1          MNEGL   #1, -(SP)
                    07  11 000C4          BRB     14$
            50  008F  C5  9E 000C6 13$:     MOVAB   P.ABB, R0                               ; 0854
                    50  DD 000CB          PUSHL   R0
                  4B   A5  9F 000CD 14$:     PUSHAB  P.AAU                                   ; 0838
```

```
                                66              02  FB 000D0           CALLS   #2, DBG$PRINT                                                ;
                                50              67  D0 000D3           MOVL    DBG$GB_STP_PTR, R0                                           ; 0863
                                          04    A0  D5 000D6           TSTL    4(R0)                                                        ;
                                                3E  13 000D9           BEQL    18$                                                          ;
                                54        04    A0  D0 000DB           MOVL    4(R0), OPCODE_LIST                                           ; 0872
                                52              01  CE 000DF           MNEGL   #1, COLUMN                                                   ; 0873
                                                53  D4 000E2           CLRL    OPCODE                                                       ; 0875
                           29                   64  E1 000E4 15$:      BBC     OPCODE, (OPCODE_LIST), 17$                                   ;
                                50         01   A2  9E 000E8           MOVAB   1(R2), R0                                                    ; 0879
                      52        50              03  EF 000EC           EXTZV   #0, #3, R0, COLUMN                                           ;
                                                03  12 000F1           BNEQ    16$                                                          ;
                                68              00  FB 000F3           CALLS   #0, DBG$NEWLINE                                              ;
                                50 00000000G0043    3C 000F6 16$:      MOVZWL  DBG$OPCODE_KIND_TABLE[OPCODE], INDEX                         ; 0880
                                50              0A  C4 000FE           MULL2   #10, R0                                                      ; 0881
                                   00000000G0040    9F 00101           PUSHAB  DBG$OPCODE_NAME_TABLE[R0]                                    ;
                                                06  DD 00108           PUSHL   #6                                                           ;
                                         0096   C5  9F 0010A           PUSHAB  P.ABC                                                        ;
                                66              03  FB 0010E           CALLS   #3, DBG$PRINT                                                ;
                           CB   53 000001FF     8F  F3 00111 17$:      AOBLEQ  #511, OPCODE, 15$                                            ; 0875
                                68              00  FB 00119 18$:      CALLS   #0, DBG$NEWLINE                                              ; 0890
                                                04  00011C            RET                                                                  ; 0891

; Routine Size:  285 bytes,    Routine Base:  DBG$CODE + 0236
```

```
;  770          0892  1 GLOBAL ROUTINE dbg$init_search (goal_level, source_level) : NOVALUE =
;  771          0893  1 !++
;  772          0894  1 ! FUNCTIONAL DESCRIPTION:
;  773          0895  1 !       Sets all levels from override level to the goal level specified
;  774          0896  1 !       with the search settings of the source level.
;  775          0897  1 !
;  776          0898  1 !
;  777          0899  1 ! INPUTS:
;  778          0900  1 !       goal_level      - highest level to set
;  779          0901  1 !       source_level    - level from which to obtain search settings
;  780          0902  1 !
;  781          0903  1 ! IMPLICIT INPUTS:
;  782          0904  1 !       none
;  783          0905  1 !
;  784          0906  1 ! OUTPUTS:
;  785          0907  1 !       NONE
;  786          0908  1 !
;  787          0909  1 ! IMPLICIT OUTPUTS:
;  788          0910  1 !       THE search settings of the override level to the goal level
;  789          0911  1 !       are reset to the search settings of the source level.
;  790          0912  1 !
;  791          0913  1 ! Routine value:
;  792          0914  1 !       NOVALUE
;  793          0915  1 !
;  794          0916  1 ! SIDE EFFECTS:
;  795          0917  1 !       NONE
;  796          0918  1 !--
;  797          0919  1
;  798          0920  2         BEGIN
;  799          0921  2
;  800          0922  2         LOCAL
;  801          0923  2                 temp_level: REF VECTOR [, BYTE];
;  802          0924  2
;  803          0925  2         temp_level = dbg$gb_def_search [.source_level * search_lvl_size];
;  804          0926  2         DECR I FROM override_search TO .goal_level DO
;  805          0927  3                 BEGIN
;  806          0928  3                 dbg$gb_search_ptr = dbg$gb_def_search [.I * search_lvl_size];
;  807          0929  3                 dbg$gb_search_ptr [search_all] = .temp_level [search_all];
;  808          0930  3                 dbg$gb_search_ptr [search_ident] = .temp_level [search_ident];
;  809          0931  2                 END;
;  810          0932  1         END;


                                001C 00000          .ENTRY  DBG$INIT_SEARCH, Save R2,R3,R4                  ; 0892
        54 00000000G   00    9E 00002          MOVAB   DBG$GB_DEF_SEARCH, R4                           ;
        53 00000000G   00    9E 00009          MOVAB   DBG$GB_SEARCH_PTR, R3                           ;
        50         08    AC  D0 00010          MOVL    SOURCE_LEVEL, R0                                ; 0925
        52             6440  3E 00014          MOVAW   DBG$GB_DEF_SEARCH[R0], TEMP_LEVEL               ;
        51               02  D0 00018          MOVL    #2, I                                           ; 0930
                         0C  11 0001B          BRB     2$                                              ;
        63             6441  3E 0001D 1$:      MOVAW   DBG$GB_DEF_SEARCH[I], DBG$GB_SEARCH_PTR         ; 0928
        50               63  D0 00021          MOVL    DBG$GB_SEARCH_PTR, R0                           ; 0929
        60               62  B0 00024          MOVW    (TEMP_LEVEL), (R0)                              ;
                         51  D7 00027          DECL    I                                               ; 0926
```

```
                    04    AC          51  D1 00029 2$:    CMPL    I, GOAL_LEVEL                   ;
                                      EE  18 0002D        BGEQ    1$                              ;
                                          04 0002F        RET                                     ; 0932

; Routine Size:  48 bytes,    Routine Base:  DBG$CODE + 0353
```

```
;  812          0933  1 GLOBAL ROUTINE dbg$set_search_def : NOVALUE =
;  813          0934  1 !++
;  814          0935  1 ! FUNCTIONAL DESCRIPTION:
;  815          0936  1 !       CANCELS USER SET DEFAULTS FOR SEARCH SETTINGS.
;  816          0937  1 !       RESETS ALL SEARCH LEVELS TO SYSTEM DEFAULTS FOR THE CURRENT LANGUAGE.
;  817          0938  1 !
;  818          0939  1 !
;  819          0940  1 ! INPUTS:
;  820          0941  1 !       NONE
;  821          0942  1 !
;  822          0943  1 ! IMPLICIT INPUTS:
;  823          0944  1 !       THE DEFAULT SEARCH SETTINGS
;  824          0945  1 !
;  825          0946  1 ! OUTPUTS:
;  826          0947  1 !       NONE
;  827          0948  1 !
;  828          0949  1 ! IMPLICIT OUTPUTS:
;  829          0950  1 !       NONE
;  830          0951  1 !
;  831          0952  1 ! Routine value:
;  832          0953  1 !       NOVALUE
;  833          0954  1 !
;  834          0955  1 ! SIDE EFFECTS:
;  835          0956  1 !       RESETS DEFAULT values FOR SEARCH.
;  836          0957  1 !--
;  837          0958  1
;  838          0959  2         BEGIN
;  839          0960  2
;  840          0961  2
;  841          0962  2         !+
;  842          0963  2         ! Set up the search data structure
;  843          0964  2         !-
;  844          0965  2
;  845          0966  2         dbg$gb_search_ptr = dbg$gb_def_search[default_search * search_lvl_size];
;  846          0967  2
;  847          0968  2         dbg$gb_search_ptr[search_all] = FALSE;
;  848          0969  2         dbg$gb_search_ptr[search_ident] = FALSE;
;  849          0970  2         dbg$init_search (user_def_search, default_search);
;  850          0971  1         END;


                                  0004 00000      .ENTRY  DBG$SET_SEARCH_DEF, Save R2                             ; 0933
                52 00000000G   00   9E 00002      MOVAB   DBG$GB_SEARCH_PTR, R2                                   ;
                62 00000000G   00   9E 00009      MOVAB   DBG$GB_DEF_SEARCH, DBG$GB_SEARCH_PTR                    ; 0966
                50             62   D0 00010      MOVL    DBG$GB_SEARCH_PTR, R0                                   ; 0968
                               60   B4 00013      CLRW    (R0)                                                    ;
                7E             01   7D 00015      MOVQ    #1, -(SP)                                               ; 0970
              B4   AF          02   FB 00018      CALLS   #2, DBG$INIT_SEARCH                                     ;
                                    04 0001C      RET                                                             ; 0971

; Routine Size:  29 bytes,    Routine Base:  DBG$CODE + 0383
```

```
;  852          0972  1 GLOBAL ROUTINE dbg$set_search_lvl (level) : NOVALUE =
;  853          0973  1 !++
;  854          0974  1 ! FUNCTIONAL DESCRIPTION
;  855          0975  1 !       This routine sets the pointer to the search structure to the specified
;  856          0976  1 !       level.  If the level is not user_def_search the search values there are
;  857          0977  1 !       copied to override_search level.  If the level is user_def_search the
;  858          0978  1 !       pointer is just reset to that level.
;  859          0979  1 !
;  860          0980  1 ! FORMAL PARAMETERS:
;  861          0981  1 !       LEVEL   - The new search structure level.
;  862          0982  1 !                 user_def_search
;  863          0983  1 !                 override_search
;  864          0984  1 !
;  865          0985  1 ! IMPLICIT INPUTS:
;  866          0986  1 !       NONE
;  867          0987  1 !
;  868          0988  1 ! IMPLICIT OUTPUTS:
;  869          0989  1 !       NONE
;  870          0990  1 !
;  871          0991  1 ! Routine value:
;  872          0992  1 !       NO value
;  873          0993  1 !
;  874          0994  1 ! SIDE EFFECTS:
;  875          0995  1 !       NONE
;  876          0996  1 !--
;  877          0997  1
;  878          0998  2         BEGIN
;  879          0999  2
;  880          1000  2         LOCAL
;  881          1001  2                 temp_level: REF VECTOR [, BYTE];
;  882          1002  2
;  883          1003  2         IF .level EQL user_def_search
;  884          1004  2         THEN
;  885          1005  3                 BEGIN
;  886          1006  3                 dbg$gb_search_ptr = dbg$gb_def_search [user_def_search * search_lvl_size];
;  887          1007  3                 END
;  888          1008  2         ELSE
;  889          1009  3                 BEGIN
;  890          1010  3                 temp_level = dbg$gb_def_search [user_def_search * search_lvl_size];
;  891          1011  3                 dbg$gb_search_ptr = dbg$gb_def_search [override_search * search_lvl_size];
;  892          1012  3
;  893          1013  3                 dbg$gb_search_ptr [search_all] = .temp_level [search_all];
;  894          1014  3                 dbg$gb_search_ptr [search_ident] = .temp_level [search_ident];
;  895          1015  2                 END;
;  896          1016  2
;  897          1017  1         END;


                                  000C 00000          .ENTRY  DBG$SET_SEARCH_LVL, Save R2,R3                  ; 0972
                53 00000000G   00   9E 00002          MOVAB   DBG$GB_DEF_SEARCH+2, R3                         ;
                52 00000000G   00   9E 00009          MOVAB   DBG$GB_SEARCH_PTR, R2                           ;
                01         04  AC   D1 00010          CMPL    LEVEL, #1                                       ; 1003
                                04   12 00014          BNEQ    1$                                              ;
                62             63   9E 00016          MOVAB   DBG$GB_DEF_SEARCH+2, DBG$GB_SEARCH_PTR          ; 1006
```

```
                         04 00019      RET                                                   ; 1003
     51          63      9E 0001A 1$:  MOVAB  DBG$GB_DEF_SEARCH+2, TEMP_LEVEL               ; 1010
     62      02  A3      9E 0001D      MOVAB  DBG$GB_DEF_SEARCH+4, DBG$GB_SEARCH_PTR        ; 1011
     50          62      D0 00021      MOVL   DBG$GB_SEARCH_PTR, R0                         ; 1013
     60          61      B0 00024      MOVW   (TEMP_LEVEL), (R0)                            ;
                         04 00027      RET                                                  ; 1017

; Routine Size:  40 bytes,    Routine Base:  DBG$CODE + 03A0
```

```
;  899          1018  1 GLOBAL ROUTINE dbg$show_search : NOVALUE =
;  900          1019  1 !++
;  901          1020  1 ! FUNCTIONAL DESCRIPTION:
;  902          1021  1 !       ACTION ROUTINE TO DISPLAY THE CURRENT SETTING OF THE
;  903          1022  1 !       "SEARCH TYPE" DATA STRUCTURE
;  904          1023  1 !
;  905          1024  1 ! INPUTS:
;  906          1025  1 !       NONE
;  907          1026  1 !
;  908          1027  1 ! IMPLICIT INPUTS·
;  909          1028  1 !       none.
;  910          1029  1 !
;  911          1030  1 ! OUTPUTS:
;  912          1031  1 !       NONE
;  913          1032  1 !
;  914          1033  1 ! IMPLICIT OUTPUTS:
;  915          1034  1 !       NONE
;  916          1035  1 !
;  917          1036  1 ! Routine value:
;  918          1037  1 !       NOVALUE
;  919          1038  1 !
;  920          1039  1 ! SIDE EFFECTS:
;  921          1040  1 !       THE SEARCH SETTINGS ARE DISPLAYED ON THE OUTPUT DEVICE
;  922          1041  1 !--
;  923          1042  1
;  924          1043  2 BEGIN
;  925          1044  2         LOCAL
;  926          1045  2                 all_cs          : REF VECTOR[,byte],
;  927          1046  2                 ident_cs        : REF VECTOR[,byte];
;  928          1047  2
;  929          1048  2         !+
;  930          1049  2         ! We simply put out one line describing the
;  931          1050  2         ! current search types, based on the TRUE/FALSE
;  932          1051  2         ! indicators given in the SEARCH structure.
;  933          1052  2         !-
;  934          1053  2
;  935          1054  2         ! If the search mode is not ALL, then it
;  936          1055  2         ! must be NEXT.
;  937          1056  2
;  938          1057  3         if( .dbg$gb_search_ptr[ search_all ] )
;  939          1058  2         then
;  940          1059  2                 all_cs = uplit( %ascic 'all occurences' )
;  941          1060  2         else
;  942          1061  2                 all_cs = uplit( %ascic 'next occurence' );
;  943          1062  2
;  944          1063  2         ! If the search mode is not IDENT, then it
;  945          1064  2         ! must be STRING.
;  946          1065  2
;  947          1066  3         if( .dbg$gb_search_ptr[ search_ident] )
;  948          1067  2         then
;  949          1068  2                 ident_cs = uplit( %ascic 'an identifier' )
;  950          1069  2         else
;  951          1070  2                 ident_cs = uplit( %ascic 'a string');
;  952          1071  2
;  953          1072  2         ! Put out the line and return.
;  954          1073  2
;  955        P 1074  2         $fao_tt_out('search settings: search for !AC, as !AC',
```

```
;  956          1075  2                    .all_cs,.ident_cs);
;  957          1076  1 END;


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

                                                            000E5           .BLKB   3
73 65 63 6E 65 72 75 63 63 6F 20 6C 6C 61 0E    000E8 P.ABD:    .ASCII  <14>\all occurences\<0>                 ;
                                           00   000F7
65 63 6E 65 72 75 63 63 6F 20 74 78 65 6E 0E    000F8 P.ABE:    .ASCII  <14>\next occurence\<0>                 ;
                                           00   00107
00 72 65 69 66 69 74 6E 65 64 69 20 6E 61 0D    00108 P.ABF:    .ASCII  <13>\an identifier\<0><0>               ;
                                           00   00117
          00 00 00 67 6E 69 72 74 73 20 61 08    00118 P.ABG:    .ASCII  <8>\a string\<0><0><0>                  ;
                                           27   00124 P.ABH:    .BYTE   39
73 67 6E 69 74 74 65 73 20 68 63 72 61 65 73    00125           .ASCII  \search settings: search for !AC, as !AC\ ;
41 21 20 72 6F 66 20 68 63 72 61 65 73 20 3A    00134                                                           ;
                     43 41 21 20 73 61 20 2C    00143                                                           ;


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

                                                    0004 00000      .ENTRY  DBG$$SHOW_SEARCH, Save R2               ; 1018
                                        52 00000000'  EF  9E 00002  MOVAB   P.ABD, R2                               ;
                                        50 00000000G  00  D0 00009  MOVL    DBG$GB_SEARCH_PTR, R0                   ; 1057
                                        05            60  E9 00010  BLBC    (R0), 1$                                ;
                                        51            62  9E 00013  MOVAB   P.ABD, ALL_CS                          ; 1059
                                                      04  11 00016  BRB     2$                                      ;
                                        51      10    A2  9E 00018  1$:    MOVAB   P.ABE, ALL_CS                   ; 1061
                                        06      01    A0  E9 0001C  2$:    BLBC    1(R0), 3$                       ; 1066
                                        50      20    A2  9E 00020        MOVAB   P.ABF, IDENT_CS                 ; 1068
                                                      04  11 00024        BRB     4$                              ;
                                        50      30    A2  9E 00026  3$:    MOVAB   P.ABG, IDENT_CS                 ; 1070
                                                      50  DD 0002A  4$:    PUSHL   IDENT_CS                        ; 1075
                                                      51  DD 0002C        PUSHL   ALL_CS                          ;
                                              3C      A2  9F 0002E        PUSHAB  P.ABH                           ;
                                    00000000G 00          03  FB 00031  CALLS   #3, DBG$FAO_OUT                  ;
                                                          04  00038  RET                                          ; 1076

; Routine Size:  57 bytes,    Routine Base:  DBG$CODE + 03C8
```

```
;  959          1077  1 GLOBAL ROUTINE dbg$init_define (goal_level, source_level) : NOVALUE =
;  960          1078  1 !++
;  961          1079  1 ! FUNCTIONAL DESCRIPTION:
;  962          1080  1 !       Sets all levels from override level to the goal level specified
;  963          1081  1 !       with the define settings of the source level.
;  964          1082  1 !
;  965          1083  1 ! INPUTS:
;  966          1084  1 !
;  967          1085  1 !       goal_level      - highest level to set
;  968          1086  1 !       source_level    - level from which to obtain define settings
;  969          1087  1 !
;  970          1088  1 ! OUTPUTS:
;  971          1089  1 !                                                                                              6
;  972          1090  1 !       NONE                                                                                   6
;  973          1091  1 !
;  974          1092  1 ! IMPLICIT OUTPUTS:
;  975          1093  1 !
;  976          1094  1 !       THE define settings of the override level to the goal level                            2
;  977          1095  1 !       are reset to the define settings of the source level.
;  978          1096  1 !
;  979          1097  1 ! Routine value:                                                                               6
;  980          1098  1 !       NOVALUE                                                                                6
;  981          1099  1 !
;  982          1100  1 ! SIDE EFFECTS:
;  983          1101  1 !       NONE
;  984          1102  1 !--                                                                                            6
;  985          1103  1 !                                                                                              6
;  986          1104  2         BEGIN
;  987          1105  2
;  988          1106  2         LOCAL
;  989          1107  2                 temp_level: REF VECTOR [, BYTE];
;  990          1108  2
;  991          1109  2         temp_level = dbg$gb_def_define [.source_level * define_lvl_size];                     2
;  992          1110  2         DECR I FROM override_define TO .goal_level DO
;  993          1111  3                 BEGIN
;  994          1112  3                 dbg$gb_define_ptr = dbg$gb_def_define [.I * define_lvl_size];
;  995          1113  3                 dbg$gb_define_ptr [define_only] = .temp_level [define_only];
;  996          1114  2                 END;
;  997          1115  1         END;


                                  001C 00000          .ENTRY  DBG$INIT_DEFINE, Save R2,R3,R4                  ; 1077
                54 00000000G   00  9E 00002          MOVAB   DBG$GB_DEF_DEFINE, R4                           ;
                53 00000000G   00  9E 00009          MOVAB   DBG$GB_DEFINE_PTR, R3                           ;
                50             64  9E 00010          MOVAB   DBG$GB_DEF_DEFINE, R0                           ; 1109
52              50       08    AC  C1 00013          ADDL3   SOURCE_LEVEL, R0, TEMP_LEVEL                    ;
                50             02  D0 00018          MOVL    #2, I                                           ; 1113
                               0C  11 0001B          BRB     2$                                              ;
63              50             54  C1 0001D 1$:      ADDL3   R4, I, DBG$GB_DEFINE_PTR                        ; 1112
                51             63  D0 00021          MOVL    DBG$GB_DEFINE_PTR, R1                           ; 1113
                61             62  90 00024          MOVB    (TEMP_LEVEL), (R1)                              ;
                               50  D7 00027          DECL    I                                               ; 1110
                  04   AC      50  D1 00029 2$:      CMPL    I, GOAL_LEVEL                                   ;
                               EE  18 0002D          BGEQ    1$                                              ;
```

```
                04 0002F        RET                                             ; 1115

; Routine Size:  48 bytes,    Routine Base:  DBG$CODE + 0401
```

```
;  999          1116  1 GLOBAL ROUTINE dbg$set_define_def : NOVALUE =
; 1000          1117  1 !++
; 1001          1118  1 ! FUNCTIONAL DESCRIPTION:
; 1002          1119  1 !       CANCELS USER SET DEFAULTS FOR define SETTINGS.
; 1003          1120  1 !       RESETS ALL define LEVELS TO SYSTEM DEFAULTS FOR THE CURRENT LANGUAGE.
; 1004          1121  1 !
; 1005          1122  1 ! INPUTS:
; 1006          1123  1 !
; 1007          1124  1 !       NONE
; 1008          1125  1 !
; 1009          1126  1 ! IMPLICIT INPUTS:
; 1010          1127  1 !
; 1011          1128  1 !       THE DEFAULT define SETTINGS
; 1012          1129  1 !
; 1013          1130  1 ! OUTPUTS:
; 1014          1131  1 !
; 1015          1132  1 !       NONE
; 1016          1133  1 !
; 1017          1134  1 ! IMPLICIT OUTPUTS:
; 1018          1135  1 !
; 1019          1136  1 !       NONE
; 1020          1137  1 !
; 1021          1138  1 ! ROUTINE VALUE:
; 1022          1139  1 !
; 1023          1140  1 !       NOVALUE
; 1024          1141  1 !
; 1025          1142  1 ! SIDE EFFECTS:
; 1026          1143  1 !
; 1027          1144  1 !       RESETS DEFAULT values FOR define.
; 1028          1145  1 !--
; 1029          1146  1
; 1030          1147  2         BEGIN
; 1031          1148  2
; 1032          1149  2
; 1033          1150  2         !+
; 1034          1151  2         ! Set up the define data structure
; 1035          1152  2         !-
; 1036          1153  2
; 1037          1154  2         dbg$gb_define_ptr = dbg$gb_def_define[default_define * define_lvl_size];
; 1038          1155  2
; 1039          1156  2         dbg$gb_define_ptr[define_only] = define_address;
; 1040          1157  2         dbg$init_define (user_def_define, default_define);
; 1041          1158  1         END;


                                  0004 00000         .ENTRY  DBG$SET_DEFINE_DEF, Save R2                     ; 1116
          52 00000000G   00   9E 00002         MOVAB   DBG$GB_DEFINE_PTR, R2                                   ;
          62 00000000G   00   9E 00009         MOVAB   DBG$GB_DEF_DEFINE, DBG$GB_DEFINE_PTR            ; 1154
          50             62   D0 00010         MOVL    DBG$GB_DEFINE_PTR, R0                           ; 1156
          60             01   90 00013         MOVB    #1, (R0)                                        ;
          7E             01   7D 00016         MOVQ    #1, -(SP)                                       ; 1157
    B3    AF             02   FB 00019         CALLS   #2, DBG$INIT_DEFINE                             ;
                              04 0001D         RET                                                     ; 1158
```

```
; Routine Size:  30 bytes,    Routine Base:  DBG$CODE + 0431
```

```
;1043     1159  1 GLOBAL ROUTINE dbg$set_define_lvl (level) : NOVALUE =
;1044     1160  1 !++
;1045     1161  1 ! FUNCTIONAL DESCRIPTION
;1046     1162  1 !       This routine sets the pointer to the define structure to the specified
;1047     1163  1 !       level.  If the level is not user_def_define the define values there are
;1048     1164  1 !       copied to override_define level.  If the level is user_def_define the
;1049     1165  1 !       pointer is just reset to that level.
;1050     1166  1 !
;1051     1167  1 ! FORMAL PARAMETERS:
;1052     1168  1 !       LEVEL   - The new define structure level.
;1053     1169  1 !                 user_def_define
;1054     1170  1 !                 override_define
;1055     1171  1 !
;1056     1172  1 ! IMPLICIT INPUTS:
;1057     1173  1 !       NONE
;1058     1174  1 !
;1059     1175  1 ! IMPLICIT OUTPUTS:
;1060     1176  1 !       NONE
;1061     1177  1 !
;1062     1178  1 ! Routine value:
;1063     1179  1 !       NO value
;1064     1180  1 !
;1065     1181  1 ! SIDE EFFECTS:
;1066     1182  1 !       NONE
;1067     1183  1 !--
;1068     1184  1
;1069     1185  2         BEGIN
;1070     1186  2
;1071     1187  2         LOCAL
;1072     1188  2                 temp_level: REF VECTOR [, BYTE];
;1073     1189  2
;1074     1190  2         IF .level EQL user_def_define
;1075     1191  2         THEN
;1076     1192  3                 BEGIN
;1077     1193  3                 dbg$gb_define_ptr = dbg$gb_def_define [user_def_define * define_lvl_size];
;1078     1194  3                 END
;1079     1195  2         ELSE
;1080     1196  3                 BEGIN
;1081     1197  3                 temp_level = dbg$gb_def_define [user_def_define * define_lvl_size];
;1082     1198  3                 dbg$gb_define_ptr = dbg$gb_def_define [override_define * define_lvl_size];
;1083     1199  3
;1084     1200  3                 dbg$gb_define_ptr [define_only] = .temp_level [define_only];
;1085     1201  2                 END;
;1086     1202  2
;1087     1203  1         END;


                                 000C 00000         .ENTRY  DBG$SET_DEFINE_LVL, Save R2,R3                  ;1159
53 00000000G   00    9E 00002     MOVAB   DBG$GB_DEF_DEFINE+1, R3                                         ;
52 00000000G   00    9E 00009     MOVAB   DBG$GB_DEFINE_PTR, R2                                           ;
01         04    AC    D1 00010     CMPL    LEVEL, #1                                                   ;1190
                 04    12 00014     BNEQ    1$                                                          ;
62               63    9E 00016     MOVAB   DBG$GB_DEF_DEFINE+1, DBG$GB_DEFINE_PTR                      ;1193
                       04 00019     RET                                                                 ;1190
```

```
                51          63  9E 0001A 1$:    MOVAB   DBG$GB_DEF_DEFINE+1, TEMP_LEVEL                 ; 1197
                62       01 A3  9E 0001D         MOVAB   DBG$GB_DEF_DEFINE+2, DBG$GB_DEFINE_PTR          ; 1198
                50          62  D0 00021         MOVL    DBG$GB_DEFINE_PTR, R0                           ; 1200
                60          61  90 00024         MOVB    (TEMP_LEVEL), (R0)                              ;
                                04 00027         RET                                                     ; 1203

; Routine Size:  40 bytes,    Routine Base:  DBG$CODE + 044F
```

```
;  1089          1204  1 GLOBAL ROUTINE dbg$show_define : NOVALUE =
;  1090          1205  1 !++
;  1091          1206  1 ! FUNCTIONAL DESCRIPTION:
;  1092          1207  1 !       ACTION ROUTINE TO DISPLAY THE CURRENT SETTING OF THE
;  1093          1208  1 !       "define TYPE" DATA STRUCTURE
;  1094          1209  1 !
;  1095          1210  1 ! INPUTS:
;  1096          1211  1 !       NONE
;  1097          1212  1 !
;  1098          1213  1 ! IMPLICIT INPUTS·
;  1099          1214  1 !       none.
;  1100          1215  1 !
;  1101          1216  1 ! OUTPUTS:
;  1102          1217  1 !       NONE
;  1103          1218  1 !
;  1104          1219  1 ! IMPLICIT OUTPUTS:
;  1105          1220  1 !       NONE
;  1106          1221  1 !
;  1107          1222  1 ! Routine value:
;  1108          1223  1 !       NOVALUE
;  1109          1224  1 !
;  1110          1225  1 ! SIDE EFFECTS:
;  1111          1226  1 !       THE define SETTINGS ARE DISPLAYED ON THE OUTPUT DEVICE
;  1112          1227  1 !--
;  1113          1228  1
;  1114          1229  2 BEGIN
;  1115          1230  2
;  1116          1231  2         LOCAL
;  1117          1232  2             def_cs : REF VECTOR [,BYTE];
;  1118          1233  2
;  1119          1234  2         !+
;  1120          1235  2         ! We simply put out one line describing the
;  1121          1236  2         ! current define types, based on the values
;  1122          1237  2         ! given in the define structure.
;  1123          1238  2         !-
;  1124          1239  2
;  1125          1240  2         CASE .dbg$gb_define_ptr [define_only]
;  1126          1241  2                 FROM define_lowest TO define_highest OF
;  1127          1242  2             SET
;  1128          1243  2
;  1129          1244  2             [define_address] :
;  1130          1245  2                 def_cs = UPLIT (%ASCIC 'ADDRESS');
;  1131          1246  2
;  1132          1247  2             [define_command] :
;  1133          1248  2                 def_cs = UPLIT (%ASCIC 'COMMAND');
;  1134          1249  2
;  1135          1250  2             [define_procedure] :
;  1136          1251  2                 def_cs = UPLIT (%ASCIC 'PROCEDURE');
;  1137          1252  2
;  1138          1253  2             [define_string] :
;  1139          1254  2                 def_cs = UPLIT (%ASCIC 'STRING');
;  1140          1255  2
;  1141          1256  2             [define_value] :
;  1142          1257  2                 def_cs = UPLIT (%ASCIC 'VALUE');
;  1143          1258  2
;  1144          1259  2             [INRANGE, OUTRANGE] :
;  1145          1260  2                 $DBG_ERROR('DBGMOD\DBG$SHOW_DEFINE');
```

```
;  1146          1261  2
;  1147          1262  2              TES;
;  1148          1263  2
;  1149          1264  2          $fao_tt_out('current setting is: DEFINE/!AC',.def_cs);
;  1150          1265  1 END;


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

                                        53  53  45  52  44  44  41  07  0014C P.ABI:   .ASCII  <7>\ADDRESS\                                    ;
                                        44  4E  41  4D  4D  4F  43  07  00154 P.ABJ:   .ASCII  <7>\COMMAND\                                    ;
                        00  00  45  52  55  44  45  43  4F  52  50  09  0015C P.ABK:   .ASCII  <9>\PROCEDURE\<0><0>                            ;
                                        00  47  4E  49  52  54  53  06  00168 P.ABL:   .ASCII  <6>\STRING\<0>                                 ;
                                        00  00  45  55  4C  41  56  05  00170 P.ABM:   .ASCII  <5>\VALUE\<0><0>                               ;
4F  48  53  24  47  42  44              5C  44  4F  4D  47  42  44  16  00178 P.ABN:   .ASCII  <22>\DBGMOD\<92>\DBG$SHOW_DEFINE\             ;
                                        45  4E  49  46  45  44  5F  57  00187                                                               ;
                                                                    1E  0018F P.ABO:   .BYTE   30                                           ;
67  6E  69  74  74  65  73  20  74  6E  65  72  72  75  63  00190                      .ASCII  \current setting is: DEFINE/!AC\          ;
43  41  21  2F  45  4E  49  46  45  44  20  3A  73  69  20  0019F                                                                           ;


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

                                                            0004 00000          .ENTRY  DBG$SHOW_DEFINE, Save R2                        ; 1204
                                                52 00000000'  EF  9E 00002      MOVAB   P.ABN, R2                                       ;
                                                50 00000000G  00  D0 00009      MOVL    DBG$GB_DEFINE_PTR, R0                           ; 1240
                                                              01  60  8F 00010      CASEB   (R0), #1, #6                                    ;
                        0033                          002D      0027      0021      00014 1$:     .WORD   3$-1$,-                         ;
                                                      000E      000E      0039      0001C                 4$-1$,-                         ;
                                                                                                        5$-1$,-                         ;
                                                                                                        6$-1$,-                         ;
                                                                                                        7$-1$,-                         ;
                                                                                                        2$-1$,-                         ;
                                                                                                        2$-1$                           ;
                                                              52  DD 00022 2$:      PUSHL   R2                                          ; 1260
                                                              01  DD 00024          PUSHL   #1                                          ;
                                                00028362      8F  DD 00026          PUSHL   #164706                                     ;
                                00000000G  00                 03  FB 0002C          CALLS   #3, LIB$SIGNAL                              ;
                                                              1C  11 00033          BRB     8$                                          ;
                                        50        D4          A2  9E 00035 3$:      MOVAB   P.ABI, DEF_CS                               ; 1245
                                                              16  11 00039          BRB     8$                                          ;
                                        50        DC          A2  9E 0003B 4$:      MOVAB   P.ABJ, DEF_CS                               ; 1248
                                                              10  11 0003F          BRB     8$                                          ;
                                        50        E4          A2  9E 00041 5$:      MOVAB   P.ABK, DEF_CS                               ; 1251
                                                              0A  11 00045          BRB     8$                                          ;
                                        50        F0          A2  9E 00047 6$:      MOVAB   P.ABL, DEF_CS                               ; 1254
                                                              04  11 0004B          BRB     8$                                          ;
                                        50        F8          A2  9E 0004D 7$:      MOVAB   P.ABM, DEF_CS                               ; 1257
                                                              50  DD 00051 8$:      PUSHL   DEF_CS                                      ; 1264
                                                      17      A2  9F 00053          PUSHAB  P.ABO                                       ;
                                00000000G  00                 02  FB 00056          CALLS   #2, DBG$FAO_OUT                             ;
                                                              04  0005D          RET                                                 ; 1265

; Routine Size:  94 bytes,    Routine Base:  DBG$CODE + 0477
```

```
;  1152          1266  1 GLOBAL ROUTINE dbg$set_out_def : NOVALUE =
;  1153          1267  1 !++
;  1154          1268  1 ! FUNCTIONAL DESCRIPTION:
;  1155          1269  1 !       Sets OUTPUT configuration to default settings. Is called from
;  1156          1270  1 !       DBG$INIT_DEBUG as part of the once only initialization phase.
;  1157          1271  1 !       Also initializes the default LOG filespec and the string buffers and
;  1158          1272  1 !       location of the LOG file's NAM block.
;  1159          1273  1 !
;  1160          1274  1 ! FORMAL PARAMETERS:
;  1161          1275  1 !       none
;  1162          1276  1 !
;  1163          1277  1 ! IMPLICIT INPUTS:
;  1164          1278  1 !       The default modes
;  1165          1279  1 !
;  1166          1280  1 ! IMPLICIT OUTPUTS:
;  1167          1281  1 !       The vector dbg$gb_def_out is initialized
;  1168          1282  1 !
;  1169          1283  1 ! ROUTINE VALUE:
;  1170          1284  1 !       none
;  1171          1285  1 !
;  1172          1286  1 ! SIDE EFFECTS:
;  1173          1287  1 !       none
;  1174          1288  1 !--
;  1175          1289  1
;  1176          1290  2         BEGIN
;  1177          1291  2
;  1178          1292  2
;  1179          1293  2         dbg$gb_def_out [out_log]        = FALSE;        ! nolog
;  1180          1294  2         dbg$gb_def_out [out_term]       = TRUE;         ! terminal
;  1181          1295  2         dbg$gb_def_out [out_verify]     = FALSE;        ! noverify
;  1182          1296  2
;  1183          1297  2         ! This variable is used to hold pointer to a user defined filespec
;  1184          1298  2
;  1185          1299  2         dbg$gl_log_buf = 0;
;  1186          1300  2
;  1187          1301  2         ! initialize default log file spec
;  1188          1302  2
;  1189          1303  2         dbg$gl_logfab [fab$l_dna] = deflog_name;
;  1190          1304  2         dbg$gl_logfab [fab$b_dns] = deflog_size;
;  1191          1305  2
;  1192          1306  2         ! Initialize address of related NAM block and the resultant string buffers
;  1193          1307  2
;  1194          1308  2         dbg$gl_lognam = dbg$get_memory((nam$c_bln+3)/%UPVAL);
;  1195          1309  2         $nam_init (nam=.dbg$gl_lognam);
;  1196          1310  2         dbg$gb_logfse = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
;  1197          1311  2         dbg$gb_logfsr = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
;  1198          1312  2         dbg$gl_logfab [fab$l_nam] = .dbg$gl_lognam;
;  1199          1313  2         dbg$gl_lognam [nam$l_esa] = .dbg$gb_logfse;
;  1200          1314  2         dbg$gl_lognam [nam$l_rsa] = .dbg$gb_logfsr;
;  1201          1315  2         dbg$gl_lognam [nam$b_ess] = nam$c_maxrss;
;  1202          1316  2         dbg$gl_lognam [nam$b_rss] = nam$c_maxrss;
;  1203          1317  2
;  1204          1318  1         END;
```

```
                                                 0FFC 00000        .ENTRY  DBG$SET_OUT_DEF, Save R2,R3,R4,R5,R6,R7,R8,-;  1266
                                                                           R9,R10,R11
                                          5B 00000000G   00  9E 00002  MOVAB   DBG$GB_LOGFSR, R11
                                          5A 00000000G   00  9E 00009  MOVAB   DBG$GB_LOGFSE, R10
                                          59 00000000G   00  9E 00010  MOVAB   DBG$GL_LOGNAM, R9
                                          58 00000000G   00  9E 00017  MOVAB   DBG$GET_MEMORY, R8
                                          57 00000000G   00  9E 0001E  MOVAB   DBG$GL_LOGFAB+48, R7
                                00000000G 00      0100   8F  B0 00025  MOVW    #256, DBG$GB_DEF_OUT                      1243
                                          00000000G   00  94 0002E  CLRB    DBG$GB_DEF_OUT+2                          1295
                                          00000000G   00  D4 00034  CLRL    DBG$GL_LOG_BUF                            1299
                                          67 00000000'   EF  9E 0003A  MOVAB   DEFLOG_NAME, DBG$GL_LOGFAB+48           1303
                                    05    A7             09  90 00041  MOVB    #9, DBG$GL_LOGFAB+53                      1304
                                                         18  DD 00045  PUSHL   #24                                       1308
                                          68             01  FB 00047  CALLS   #1, DBG$GET_MEMORY
                                          69             50  D0 0004A  MOVL    R0, DBG$GL_LOGNAM
                                          56             69  D0 0004D  MOVL    DBG$GL_LOGNAM, R6                         1309
                0060    8F          00                 6E  00  2C 00050  MOVCS   #0, (SP), #0, #96, (R6)
                                                             66  00057
                                          66       6002  8F  B0 00058  MOVW    #24578, (R6)
                                          7E         40  8F  9A 0005D  MOVZBL  #64, -(SP)                               1310
                                          68             01  FB 00061  CALLS   #1, DBG$GET_MEMORY
                                          6A             50  D0 00064  MOVL    R0, DBG$GB_LOGFSE
                                          7E         40  8F  9A 00067  MOVZBL  #64, -(SP)                               1311
                                          68             01  FB 0006B  CALLS   #1, DBG$GET_MEMORY
                                          6B             50  D0 0006E  MOVL    R0, DBG$GB_LOGFSR
                                          50             69  D0 00071  MOVL    DBG$GL_LOGNAM, R0                         1312
                                    F8    A7             50  D0 00074  MOVL    R0, DBG$GL_LOGFAB+40
                                    0C    A0             6A  D0 00078  MOVL    DBG$GB_LOGFSE, 12(R0)                     1313
                                    04    A0             6B  D0 0007C  MOVL    DBG$GB_LOGFSR, 4(R0)                      1314
                                    0A    A0             01  8E 00080  MNEGB   #1, 10(R0)                              1315
                                    02    A0             01  8E 00084  MNEGB   #1, 2(R0)                               1316
                                                         04  00088  RET                                               1318

; Routine Size:  137 bytes,    Routine Base:  DBG$CODE + 04D5
```

```
;  1206          1319  1 GLOBAL ROUTINE dbg$show_output(full_rep) : NOVALUE =
;  1207          1320  1 !++
;  1208          1321  1 ! FUNCTIONAL DESCRIPTION:
;  1209          1322  1 !       Action routine to sh    JTPUT configuration
;  1210          1323  1 !
;  1211          1324  1 ! FORMAL PARAMETERS:
;  1212          1325  1 !       full_rep    equals 1 for a full report (i.e. "SHOW OUTPUT")
;  1213          1326  1 !                          2 for just "SHOW LOG"
;  1214          1327  1 !
;  1215          1328  1 ! IMPLICIT INPUTS:
;  1216          1329  1 !       The OUTPUT configuration data structure, dbg$gb_def_out
;  1217          1330  1 !
;  1218          1331  1 ! IMPLICIT OUTPUTS:
;  1219          1332  1 !       none
;  1220          1333  1 !
;  1221          1334  1 ! ROUTINE VALUE:
;  1222          1335  1 !       none
;  1223          1336  1 !
;  1224          1337  1 ! SIDE EFFECTS:
;  1225          1338  1 !       A message is generated that displays the current output configuration
;  1226          1339  1 !       and the log file name
;  1227          1340  1 !--
;  1228          1341  1
;  1229          1342  2         BEGIN
;  1230          1343  2
;  1231          1344  2         LOCAL
;  1232          1345  2                 verify_cs :     REF VECTOR [,BYTE],
;  1233          1346  2                 term_cs :       REF VECTOR [,BYTE],
;  1234          1347  2                 log_cs :        REF VECTOR [,BYTE],
;  1235          1348  2                 no_count:       word,
;  1236          1349  2                 string_count:   word;                   ! count of chars in output string
;  1237          1350  2
;  1238          1351  2         no_count = 0;                           ! count of "no" characters
;  1239          1352  2
;  1240          1353  2         IF .full_rep
;  1241          1354  2         THEN
;  1242          1355  3                 BEGIN
;  1243          1356  3                 IF .dbg$gb_def_out [out_verify]
;  1244          1357  3                 THEN
;  1245          1358  3                         verify_cs = UPLIT(%ASCIC '')
;  1246          1359  3                 ELSE
;  1247          1360  4                         BEGIN
;  1248          1361  4                         verify_cs = UPLIT(%ASCIC 'no');
;  1249          1362  4                         no_count = .no_count + 2;
;  1250          1363  3                         END;
;  1251          1364  3
;  1252          1365  3                 IF .dbg$gb_def_out [out_term]
;  1253          1366  3                 THEN
;  1254          1367  3                         term_cs = UPLIT(%ASCIC '')
;  1255          1368  3                 ELSE
;  1256          1369  4                         BEGIN
;  1257          1370  4                         term_cs = UPLIT(%ASCIC 'no');
;  1258          1371  4                         no_count = .no_count + 2
;  1259          1372  3                         END;
;  1260          1373  2                 END;
;  1261          1374  2
;  1262          1375  2         IF .dbg$gb_def_out [out_log]
```

```
;  1263          1376  2          THEN
;  1264          1377  2                  log_cs = UPLIT(%ASCIC '')
;  1265          1378  2          ELSE
;  1266          1379  3                  BEGIN
;  1267          1380  3                  log_cs = UPLIT(%ASCIC 'not ');
;  1268          1381  3                  no_count = .no_count + 3;
;  1269          1382  2                  END;
;  1270          1383  2
;  1271        L 1384  2 %IF dbg_mod2
;  1272        U 1385  2 %THEN
;  1273        U 1386  2 $fao_tt_out('show_output: rss !SB, rsa !XL',,.dbg$gl_lognam[nam$b_rss], .dbg$gl_lognam[nam$l_rsa]);
;  1274          1387  2 %FI
;  1275          1388  2
;  1276          1389  2          ! If a LOG file is open report the filespec in the NAM block, otherwise
;  1277          1390  2          ! use the default filespec
;  1278          1391  2
;  1279          1392  2
;  1280          1393  2          IF .dbg$gl_logfab [fab$w_ifi] LEQ 0
;  1281          1394  2          THEN
;  1282          1395  2                  IF .full_rep
;  1283          1396  2                  THEN
;  1284        P 1397  2                  $fao_tt_out('output: !ACverify, !ACterminal, !AClogging to !AD',
;  1285          1398  2                  .verify_cs, .term_cs, .log_cs, deflog_size, deflog_name)
;  1286          1399  2                  ELSE
;  1287          1400  2                  $fao_tt_out('!AClogging to !AD',,.log_cs, deflog_size, deflog_name)
;  1288          1401  2
;  1289          1402  2          ELSE
;  1290          1403  2              IF .full_rep
;  1291          1404  2              THEN
;  1292          1405  3              BEGIN
;  1293          1406  3              ! string count is initialized to the no. of characters in the output
;  1294          1407  3              ! string (not counting "no"s and the log filespec).
;  1295          1408  3
;  1296          1409  3              string_count = 38;
;  1297          1410  3              string_count = .string_count + .no_count + .dbg$gl_lognam[nam$b_rsl];
;  1298          1411  3
;  1299          1412  3              ! If we exceed the max record length, output two lines
;  1300          1413  3
;  1301          1414  3              IF .string_count LEQ tty_out_width
;  1302          1415  3              THEN
;  1303        P 1416  3              $fao_tt_out('output: !ACverify, !ACterminal, !AClogging to !AD',
;  1304          1417  3              .verify_cs, .term_cs, .log_cs, .dbg$gl_lognam[nam$b_rsl], .dbg$gl_lognam[nam$l_rsa])
;  1305          1418  3
;  1306          1419  3              ELSE
;  1307          1420  4                  BEGIN
;  1308        P 1421  4                  $fao_tt_out('output: !ACverify, !ACterminal, !AClogging to :',
;  1309          1422  4                          .verify_cs, .term_cs, .log_cs);
;  1310          1423  4                  $fao_tt_out('!AD', .dbg$gl_lognam[nam$b_rsl], .dbg$gl_lognam[nam$l_rsa]);
;  1311          1424  3                  END;
;  1312          1425  3              END
;  1313          1426  3
;  1314          1427  2              ELSE
;  1315        P 1428  2              $fao_tt_out('!AClogging to !AD', .log_cs, .dbg$gl_lognam[nam$b_rsl],
;  1316          1429  2                          .dbg$gl_lognam[nam$l_rsa]);
;  1317          1430  2
;  1318          1431  1      END;
```

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

                                                                                001AE           .BLKB   2
                                                                        00      001B0 P.ABP:    .ASCII  <0><0><0><0>
                                                                        00      001B4 P.ABQ:    .ASCII  <2>\no\<0>
                                                                        00      001B8 P.ABR:    .ASCII  <0><0><0><0>
                                                                        00      001BC P.ABS:    .ASCII  <2>\no\<0>
                                                                        00      001C0 P.ABT:    .ASCII  <0><0><0><0>
                                                        00      00      04      001C4 P.ABU:    .ASCII  <4>\not \<0><0><0>
                                                                        31      001CC P.ABV:    .BYTE   49
69  72  65  76  43  41  21  20  3A  74  75  70  74  75  6F                      001CD           .ASCII  \output: !ACverify, !ACterminal, !ACloggi\
6C  61  6E  69  6D  72  65  74  43  41  21  20  2C  79  66                      001DC
                    69  67  67  6F  6C  43  41  21  20  2C                      001EB
                        44  41  21  20  6F  74  20  67  6E                      001F5           .ASCII  \ng to !AD\
                                                                        11      001FE P.ABW:    .BYTE   17
21  20  6F  74  20  67  6E  69  67  67  6F  6C  43  41  21                      001FF           .ASCII  \!AClogging to !AD\
                                                    44  41                      0020E
                                                                        31      00210 P.ABX:    .BYTE   49
69  72  65  76  43  41  21  20  3A  74  75  70  74  75  6F                      00211           .ASCII  \output: !ACverify, !ACterminal, !ACloggi\
6C  61  6E  69  6D  72  65  74  43  41  21  20  2C  79  66                      00220
                    69  67  67  6F  6C  43  41  21  20  2C                      0022F
                        44  41  21  20  6F  74  20  67  6E                      00239           .ASCII  \ng to !AD\
                                                                        2F      00242 P.ABY:    .BYTE   47
69  72  65  76  43  41  21  20  3A  74  75  70  74  75  6F                      00243           .ASCII  \output: !ACverify, !ACterminal, !ACloggi\
6C  61  6E  69  6D  72  65  74  43  41  21  20  2C  79  66                      00252
                    69  67  67  6F  6C  43  41  21  20  2C                      00261
                            3A  20  6F  74  20  67  6E                      0026B           .ASCII  \ng to :\
                                                                        03      00272 P.ABZ:    .BYTE   3
                                                        44  41  21          00273           .ASCII  \!AD\
                                                                        11      00276 P.ACA:    .BYTE   17
21  20  6F  74  20  67  6E  69  67  67  6F  6C  43  41  21                      00277           .ASCII  \!AClogging to !AD\
                                                    44  41                      00286


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

                                                                        0FFC    00000          .ENTRY  DBG$SHOW_OUTPUT, Save R2,R3,R4,R5,R6,R7,R8,-;  1319
                                                                                                       R9,R10,R11
                                                5B  00000000G   00      9E      00002          MOVAB   DBG$GL_LOGNAM, R11
                                                5A  00000000G   00      9E      00009          MOVAB   DBG$GB_DEF_OUT+2, R10
                                                59  00000000G   00      9E      00010          MOVAB   DBG$FAO_OUT, R9
                                                58  00000000'   EF      9E      00017          MOVAB   DEFLOG_NAME, R8
                                                                54      B4      0001E          CLRW    NO_COUNT                                1351
                                                25          04  AC      E9      00020          BLBC    FULL_REP, 4$                            1353
                                                07              6A      E9      00024          BLBC    DBG$GB_DEF_OUT+2, 1$                    1356
                                                56      01B0    C8      9E      00027          MOVAB   P.ABP, VERIFY_CS                       1358
                                                                08      11      0002C          BRB     2$
                                                56      01B4    C8      9E      0002E  1$:     MOVAB   P.ABQ, VERIFY_CS                       1361
                                                54              02      A0      00033          ADDW2   #2, NO_COUNT                            1362
                                                07          FF  AA      E9      00036  2$:     BLBC    DBG$GB_DEF_OUT+1, 3$                    1365
                                                55      01B8    C8      9E      0003A          MOVAB   P.ABR, TERM_CS                          1367
                                                                08      11      0003F          BRB     4$
                                                55      01BC    C8      9E      00041  3$:     MOVAB   P.ABS, TERM_CS                         1370
                                                54              02      A0      00046          ADDW2   #2, NO_COUNT                            1371
```

```
                07      FE    AA  E9 00049 4$:     BLBC    DBG$GB_DEF_OUT, 5$                      ; 1375
                53    01C0    C8  9E 0004D         MOVAB   P.ABT, LOG_CS                           ; 1377
                              08  11 00052         BRB     6$
                53    01C4    C8  9E 00054 5$:     MOVAB   P.ABU, LOG_CS                           ; 1380
                54            03  A0 00059         ADDW2   #3, NO_COUNT                            ; 1381
                   00000000G  00  B5 0005C 6$:     TSTW    DBG$GL_LOGFAB+2                         ; 1393
                              20  12 00062         BNEQ    8$
                10      04    AC  E9 00064         BLBC    FULL_REP, 7$                            ; 1395
                              58  DD 00068         PUSHL   R8                                      ; 1398
                              09  DD 0006A         PUSHL   #9
                              53  DD 0006C         PUSHL   LOG_CS
                              55  DD 0006E         PUSHL   TERM_CS
                              56  DD 00070         PUSHL   VERIFY_CS
                        01CC  C8  9F 00072         PUSHAB  P.ABV
                              3F  11 00076         BRB     9$
                              58  DD 00078 7$:     PUSHL   R8                                      ; 1400
                              09  DD 0007A         PUSHL   #9
                              53  DD 0007C         PUSHL   LOG_CS
                        01FE  C8  9F 0007E         PUSHAB  P.ABW
                              63  11 00082         BRB     12$
                50            6B  D0 00084 8$:     MOVL    DBG$GL_LOGNAM, R0                       ; 1410
                4F      04    AC  E9 00087         BLBC    FULL_REP, 11$
                52            26  B0 0008B         MOVW    #38, STRING_COUNT                       ; 1409
                51            52  3C 0008E         MOVZWL  STRING_COUNT, R1                        ; 1410
                57            54  3C 00091         MOVZWL  NO_COUNT, R7
                51            57  C0 00094         ADDL2   R7, R1
                54      03    A0  9A 00097         MOVZBL  3(R0), R4
        52      51            54  A1 0009B         ADDW3   R4, R1, STRING_COUNT
              0084    8F      52  B1 0009F         CMPW    STRING_COUNT, #132                      ; 1414
                              15  1A 000A4         BGTRU   10$
                      04      A0  DD 000A6         PUSHL   4(R0)                                   ; 1417
                7E    03      A0  9A 000A9         MOVZBL  3(R0), -(SP)
                              53  DD 000AD         PUSHL   LOG_CS
                              55  DD 000AF         PUSHL   TERM_CS
                              56  DD 000B1         PUSHL   VERIFY_CS
                        0210  C8  9F 000B3         PUSHAB  P.ABX
                69            06  FB 000B7 9$:     CALLS   #6, DBG$FAO_OUT
                              04  000BA           RET
                              53  DD 000BB 10$:     PUSHL   LOG_CS                                  ; 1422
                              55  DD 000BD         PUSHL   TERM_CS
                              56  DD 000BF         PUSHL   VERIFY_CS
                        0242  C8  9F 000C1         PUSHAB  P.ABY
                69            04  FB 000C5         CALLS   #4, DBG$FAO_OUT
                50            6B  D0 000C8         MOVL    DBG$GL_LOGNAM, R0                       ; 1423
                      04      A0  DD 000CB         PUSHL   4(R0)
                7E    03      A0  9A 000CE         MOVZBL  3(R0), -(SP)
                        0272  C8  9F 000D2         PUSHAB  P.ABZ
                69            03  FB 000D6         CALLS   #3, DBG$FAO_OUT
                              04  000D9           RET                                             ; 1403
                      04      A0  DD 000DA 11$:     PUSHL   4(R0)                                   ; 1429
                7E    03      A0  9A 000DD         MOVZBL  3(R0), -(SP)
                              53  DD 000E1         PUSHL   LOG_CS
                        0276  C8  9F 000E3         PUSHAB  P.ACA
                69            04  FB 000E7 12$:     CALLS   #4, DBG$FAO_OUT
                              04  000EA           RET                                             ; 1431

; Routine Size:  235 bytes,    Routine Base:  DBG$CODE + 055E
```

```
;  1320          1432  1 GLOBAL ROUTINE dbg$set_log : NOVALUE =
;  1321          1433  1 !++
;  1322          1434  1 ! FUNCTIONAL DESCRIPTION:
;  1323          1435  1 !       This routine closes any log file that is currently open and opens
;  1324          1436  1 ! the file specified by the user in SET LOG "filespec"
;  1325          1437  1 !
;  1326          1438  1 ! FORMAL PARAMETERS:
;  1327          1439  1 !       none
;  1328          1440  1 !
;  1329          1441  1 ! IMPLICIT INPUTS·
;  1330          1442  1 !       The RMS-set IFI (internal file identifier field) bit is used to
;  1331          1443  1 ! determine if a previous log file is still open.
;  1332          1444  1 !
;  1333          1445  1 ! The user specified LOG filespec is pointed to by dbg$gl_log_buf
;  1334          1446  1 !
;  1335          1447  1 ! IMPLICIT OUTPUTS:
;  1336          1448  1 !       none
;  1337          1449  1 !
;  1338          1450  1 ! ROUTINE VALUE:
;  1339          1451  1 !       none
;  1340          1452  1 !
;  1341          1453  1 ! SIDE EFFECTS:
;  1342          1454  1 !       Any previously open log file is closed and a new log file is opened.
;  1343          1455  1 ! Logging is inhibited for the duration of this routine.
;  1344          1456  1 !--
;  1345          1457  1
;  1346          1458  2         BEGIN
;  1347          1459  2
;  1348          1460  2         LOCAL
;  1349          1461  2                 status,
;  1350          1462  2                 opened_log,
;  1351          1463  2                 fns : BYTE,
;  1352          1464  2                 fna,
;  1353          1465  2                 old_nam_ptr,
;  1354          1466  2                 b_type,
;  1355          1467  2                 temp_nam : REF $NAM_DECL,
;  1356          1468  2                 temp_fse : REF VECTOR [,BYTE],
;  1357          1469  2                 temp_fsr : REF VECTOR [,BYTE],
;  1358          1470  2                 log_temp : BYTE;
;  1359          1471  2
;  1360          1472  2         opened_log = FALSE ;
;  1361          1473  2
;  1362          1474  2         ! inhibit logging until we can redirect the log file
;  1363          1475  2
;  1364          1476  2         log_temp = .dbg$gb_def_out [out_log];
;  1365          1477  2         dbg$gb_def_out [out_log] = FALSE;
;  1366          1478  2
;  1367          1479  2         ! see if a log file has been previously opened. If so, close it,
;  1368          1480  2         ! since it must be CLOSEd befor we can do the new $PARSE.
;  1369          1481  2
;  1370          1482  2         IF .dbg$gl_logfab [fab$w_ifi] NEQ 0
;  1371          1483  2         THEN
;  1372          1484  3                 BEGIN
;  1373          1485  3
;  1374          1486  3                 opened_log = TRUE;
;  1375          1487  3                 fna = .dbg$gl_logfab [fab$l_fna];
;  1376          1488  3                 fns = .dbg$gl_logfab [fab$b_fns];
```

```
;  1377          1489  3
;  1378          1490  3                    status = $CLOSE (FAB = dbg$gl_logfab);
;  1379          1491  3                    IF NOT .status
;  1380          1492  3                    THEN
;  1381          1493  4                            BEGIN
;  1382          1494  4                            LOCAL
;  1383          1495  4                                    msg_desc : BLOCK [8,BYTE];
;  1384          1496  4                            build_err_desc (msg_desc);
;  1385          1497  4                            SIGNAL (shr$_closeout+dbg_fac_code, 1, msg_desc,
;  1386          1498  4                                    .dbg$gl_logfab[fab$l_sts], .dbg$gl_logfab[fab$l_stv]);
;  1387          1499  3                            END;
;  1388          1500  3
;  1389          1501  3                    ! get new NAM block, preserve the original
;  1390          1502  3
;  1391          1503  3                    temp_nam = dbg$get_memory((nam$c_bln+3)/%UPVAL);
;  1392          1504  3                    $nam_init (nam = .temp_nam);
;  1393          1505  3                    temp_fse = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
;  1394          1506  3                    temp_fsr = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
;  1395          1507  3                    temp_nam [nam$l_esa] = .temp_fse;
;  1396          1508  3                    temp_nam [nam$l_rsa] = .temp_fsr;
;  1397          1509  3                    temp_nam [nam$b_ess] = nam$c_maxrss;
;  1398          1510  3                    temp_nam [nam$b_rss] = nam$c_maxrss;
;  1399          1511  3                    dbg$gl_logfab [fab$l_nam] = .temp_nam;
;  1400          1512  3                    old_nam_ptr = .dbg$gl_lognam;
;  1401          1513  3                    dbg$gl_lognam = .temp_nam;
;  1402          1514  3
;  1403          1515  2                    END;
;  1404          1516  2
;  1405        L 1517  2 %IF dbg_mod2
;  1406        U 1518  2 %THEN
;  1407        U 1519  2 $fao_tt_out('dbg$gl_log_buf- !XL',.dbg$gl_log_buf);
;  1408          1520  2 %FI
;  1409          1521  2
;  1410          1522  2          ! reset the appropriate FAB file process options
;  1411          1523  2
;  1412          1524  2          dbg$gl_logfab [fab$v_cif] = 0;
;  1413          1525  2          dbg$gl_logfab [fab$v_mxv] = 1;
;  1414          1526  2          dbg$gl_logfab [fab$v_nam] = 0;
;  1415          1527  2
;  1416          1528  2          ! Set up file name, dbg$gl_log_buf contains the pointer to the user
;  1417          1529  2          ! specified filespec.
;  1418          1530  2
;  1419          1531  2          IF .dbg$gl_log_buf NEQ 0
;  1420          1532  2          THEN
;  1421          1533  3                  BEGIN
;  1422          1534  3
;  1423          1535  3                  MAP
;  1424          1536  3                          dbg$gl_log_buf : REF VECTOR [,BYTE];
;  1425          1537  3
;  1426          1538  3                  dbg$gl_logfab [fab$l_fna] = .dbg$gl_log_buf + 1; ! filename addr starts in 2nd byte
;  1427          1539  3                  dbg$gl_logfab [fab$b_fns] = .dbg$gl_log_buf[0];  ! 1st byte is count
;  1428        L 1540  3 %IF dbg_mod2
;  1429        U 1541  3 %THEN
;  1430        U 1542  3 $fao_tt_out('fna !AD',.dbg$gl_logfab[fab$b_fns], .dbg$gl_logfab[fab$l_fna]);
;  1431          1543  3 %FI
;  1432          1544  3
;  1433          1545  3                  ! Parse the filespec to see if an explicit version number was
```

```
;  1434          1546  3          ! given. If so set the CIF bit, otherwise we must maximize
;  1435          1547  3          ! the version number.
;  1436          1548  3
;  1437          1549  3          status = $PARSE (FAB = dbg$gl_logfab);
;  1438          1550  3          IF NOT .status
;  1439          1551  3          THEN
;  1440          1552  4                  BEGIN
;  1441          1553  4                  LOCAL
;  1442          1554  4                          sts,
;  1443          1555  4                          stv,
;  1444          1556  4                          msg_desc : BLOCK [8,BYTE];
;  1445          1557  4
;  1446          1558  4                  ! save these because restore_nam will reset them
;  1447          1559  4
;  1448          1560  4                  sts = .dbg$gl_logfab [fab$l_sts];
;  1449          1561  4                  stv = .dbg$gl_logfab [fab$l_stv];
;  1450          1562  4                  build_err_desc (msg_desc);
;  1451          1563  4
;  1452          1564  4                  IF .opened_log
;  1453          1565  4                  THEN
;  1454          1566  5                          restore_nam
;  1455          1567  4                  ELSE
;  1456          1568  5                          BEGIN
;  1457          1569  5                          dbg$gl_logfab [fab$l_fna] = 0;
;  1458          1570  5                          dbg$gl_logfab [fab$b_fns] = 0;
;  1459          1571  5                          dbg$gl_log_buf  = 0;
;  1460          1572  4                          END;
;  1461          1573  4
;  1462          1574  4                  SIGNAL (shr$_syntax + dbg_fac_code, 1, msg_desc, .sts, .stv);
;  1463          1575  3                  END;
;  1464          1576  3
;  1465        L 1577  3 %IF dbg_mod2
;  1466        U 1578  3 %THEN
;  1467        U 1579  3 $fao_tt_out('status: !XL', .status);
;  1468        U 1580  3 $fao_tt_out('nam esa- !AD',.dbg$gl_lognam[nam$b_esl], .dbg$gl_lognam[nam$l_esa]);
;  1469          1581  3 %FI
;  1470          1582  3                  IF .dbg$gl_lognam [nam$v_exp_ver]
;  1471          1583  3                  THEN
;  1472          1584  4                          BEGIN
;  1473          1585  4                          dbg$gl_logfab [fab$v_cif] = 1;
;  1474          1586  4                          dbg$gl_logfab [fab$v_mxv] = 0;
;  1475          1587  3                          END;
;  1476          1588  3
;  1477          1589  3                  dbg$gl_logfab [fab$v_nam] = 1;          ! open by NAM block since
;  1478          1590  2                  END;                                    ! we've already parsed the filespec
;  1479          1591  2
;  1480          1592  2
;  1481        L 1593  2 %IF dbg_mod2
;  1482        U 1594  2 %THEN
;  1483        U 1595  2 $fao_tt_out('fna !AD',.dbg$gl_logfab[fab$b_fns], .dbg$gl_logfab[fab$l_fna]);
;  1484          1596  2 %FI
;  1485          1597  2
;  1486          1598  2          ! now open the new log file
;  1487          1599  2
;  1488          1600  2          status = setup_log(b_type);
;  1489          1601  2          IF NOT .status
;  1490          1602  2          THEN
```

```
;  1491     1603  3                BEGIN
;  1492     1604  3                LOCAL
;  1493     1605  3                        msg_desc : BLOCK [8,BYTE],
;  1494     1606  3                        sts,
;  1495     1607  3                        stv;
;  1496     1608  3
;  1497     1609  3                build_err_desc (msg_desc);
;  1498     1610  3                IF .b_type EQL 1
;  1499     1611  3                THEN
;  1500     1612  4                        BEGIN
;  1501     1613  4                        sts = .dbg$gl_logfab [fab$l_sts];
;  1502     1614  4                        stv = .dbg$gl_logfab [fab$l_stv];
;  1503     1615  4                        END
;  1504     1616  3                ELSE
;  1505     1617  4                        BEGIN
;  1506     1618  4                        sts = .dbg$gl_lograb [rab$l_sts];
;  1507     1619  4                        stv = .dbg$gl_lograb [rab$l_stv];
;  1508     1620  3                        END;
;  1509     1621  3
;  1510     1622  3                IF .opened_log
;  1511     1623  3                THEN
;  1512     1624  4                        restore_nam
;  1513     1625  3                ELSE
;  1514     1626  4                        BEGIN
;  1515     1627  4                        dbg$gl_logfab [fab$l_fna] = 0;
;  1516     1628  4                        dbg$gl_logfab [fab$b_fns] = 0;
;  1517     1629  4                        dbg$gl_log_buf  = 0;
;  1518     1630  4                        dbg$gl_logfab [fab$v_nam] = 0;
;  1519     1631  3                        END;
;  1520     1632  3
;  1521     1633  3                SIGNAL (shr$_openout+dbg_fac_code, 1, msg_desc, .sts, .stv);
;  1522     1634  2                END;
;  1523     1635  2
;  1524     1636  2        ! restore logging status and free temp storage
;  1525     1637  2
;  1526     1638  2        dbg$gb_def_out [out_log] = .log_temp;
;  1527     1639  2        IF .opened_log
;  1528     1640  2        THEN
;  1529     1641  2
;  1530     1642  2                ! Cleanup the "original" NAM block
;  1531     1643  2
;  1532     1644  3                BEGIN
;  1533     1645  3                dbg$rel_memory (.old_nam_ptr);
;  1534     1646  3                dbg$rel_memory (.dbg$gb_logfse);
;  1535     1647  3                dbg$rel_memory (.dbg$gb_logfsr);
;  1536     1648  3                dbg$gb_logfsr = .temp_fsr;
;  1537     1649  3                dbg$gb_logfse = .temp_fse;
;  1538     1650  3
;  1539     1651  3                ! Release the buffer that holds the log file spec. Since this
;  1540     1652  3                ! is a counted string we have to adjust the address to free
;  1541     1653  3
;  1542     1654  3                IF .fns NEQ 0
;  1543     1655  3                THEN
;  1544     1656  3                        dbg$rel_memory(.fna-1);
;  1545     1657  2                END;
;  1546     1658  1        END;
```

```
                                                          .EXTRN  SYS$CLOSE, SYS$PARSE

                                         0FFC 00000        .ENTRY  DBG$SET_LOG, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1432
                                                                   R10,R11                                      ;
                           5E            20  C2 00002        SUBL2   #32, SP                                        ;
                                         59  D4 00005        CLRL    OPENED_LOG                                     ; 1472
                           5A 00000000G  00  90 00007        MOVB    DBG$GB_DEF_OUT, LOG_TEMP                       ; 1476
                              00000000G  00  94 0000E        CLRB    DBG$GB_DEF_OUT                                 ; 1477
                              00000000G  00  B5 00014        TSTW    DBG$GL_LOGFAB+2                                ; 1482
                                         03  12 0001A        BNEQ    1$                                             ;
                                       00C1  31 0001C        BRW     5$                                             ;
                           59            01  D0 0001F 1$:    MOVL    #1, OPENED_LOG                                 ; 1486
                           57 00000000G  00  D0 00022        MOVL    DBG$GL_LOGFAB+44, FNA                          ; 1487
                           6E 00000000G  00  90 00029        MOVB    DBG$GL_LOGFAB+52, FNS                          ; 1488
                              00000000G  00  9F 00030        PUSHAB  DBG$GL_LOGFAB                                  ; 1490
                00000000G  00            01  FB 00036        CALLS   #1, SYS$CLOSE                                  ;
                           58            50  D0 0003D        MOVL    R0, STATUS                                     ;
                           43            58  E8 00040        BLBS    STATUS, 4$                                     ; 1491
                              00000000G  00  D5 00043        TSTL    DBG$GL_LOG_BUF                                 ; 1496
                                         12  13 00049        BEQL    2$                                             ;
                      18   AE 00000000G  00  9B 0004B        MOVZBW  DBG$GL_LOGFAB+52, MSG_DESC                     ;
                      1C   AE 00000000G  00  D0 00053        MOVL    DBG$GL_LOGFAB+44, MSG_DESC+4                   ;
                                         10  11 0005B        BRB     3$                                             ;
                      18   AE 00000000G  00  9B 0005D 2$:    MOVZBW  DBG$GL_LOGFAB+53, MSG_DESC                     ;
                      1C   AE 00000000G  00  D0 00065        MOVL    DBG$GL_LOGFAB+48, MSG_DESC+4                   ;
                           7E 00000000G  00  7D 0006D 3$:    MOVQ    DBG$GL_LOGFAB+8, -(SP)                         ; 1498
                                     20  AE  9F 00074        PUSHAB  MSG_DESC                                       ; 1497
                                         01  DD 00077        PUSHL   #1                                             ;
                              00021058   8F  DD 00079        PUSHL   #135256                                        ;
                00000000G  00            05  FB 0007F        CALLS   #5, LIB$SIGNAL                                 ;
                                         18  DD 00086 4$:    PUSHL   #24                                            ; 1503
                00000000G  00            01  FB 00088        CALLS   #1, DBG$GET_MEMORY                             ;
                           56            50  D0 0008F        MOVL    R0, TEMP_NAM                                   ;
  0060   8F          00    6E            00  2C 00092        MOVC5   #0, (SP), #0, #96, (TEMP_NAM)                  ; 1504
                           66               00099                                                               ;
                           66       6002 8F  B0 0009A        MOVW    #24578, (TEMP_NAM)                             ;
                           7E         40 8F  9A 0009F        MOVZBL  #64, -(SP)                                     ; 1505
                00000000G  00            01  FB 000A3        CALLS   #1, DBG$GET_MEMORY                             ;
                           5B            50  D0 000AA        MOVL    R0, TEMP_FSE                                   ;
                           7E         40 8F  9A 000AD        MOVZBL  #64, -(SP)                                     ; 1506
                00000000G  00            01  FB 000B1        CALLS   #1, DBG$GET_MEMORY                             ;
                           55            50  D0 000B8        MOVL    R0, TEMP_FSR                                   ;
                      0C   A6            5B  D0 000BB        MOVL    TEMP_FSE, 12(TEMP_NAM)                         ; 1507
                      04   A6            55  D0 000BF        MOVL    TEMP_FSR, 4(TEMP_NAM)                          ; 1508
                      0A   A6            01  8E 000C3        MNEGB   #1, 10(TEMP_NAM)                               ; 1509
                      02   A6            01  8E 000C7        MNEGB   #1, 2(TEMP_NAM)                                ; 1510
                00000000G  00            56  D0 000CB        MOVL    TEMP_NAM, DBG$GL_LOGFAB+40                     ; 1511
                           54 00000000G  00  D0 000D2        MOVL    DBG$GL_LOGNAM, OLD_NAM_PTR                     ; 1512
                00000000G  00            56  D0 000D9        MOVL    TEMP_NAM, DBG$GL_LOGNAM                        ; 1513
                00000000G  00            02  8A 000E0 5$:    BICB2   #2, DBG$GL_LOGFAB+7                            ; 1524
                00000000G  00            02  88 000E7        BISB2   #2, DBG$GL_LOGFAB+4                            ; 1525
                00000000G  00            01  8A 000EE        BICB2   #1, DBG$GL_LOGFAB+7                            ; 1526
                           50 00000000G  00  D0 000F5        MOVL    DBG$GL_LOG_BUF, R0                             ; 1531
                                         03  12 000FC        BNEQ    6$                                             ;
                                       016E  31 000FE        BRW     20$                                            ;
```

```
00000000G  00        01   A0  9E 00101 6$:    MOVAB   1(R0), DBG$GL_LOGFAB+44                 ; 1538
00000000G  00             60  90 00109        MOVB    (R0), DBG$GL_LOGFAB+52                  ; 1539
               00000000G  00  9F 00110        PUSHAB  DBG$GL_LOGFAB                           ; 1549
00000000G  00             01  FB 00116        CALLS   #1, SYS$PARSE                           ;
           58             50  D0 0011D        MOVL    R0, STATUS                              ;
           03             58  E9 00120        BLBC    STATUS, 7$                              ; 1550
                        0129  31 00123        BRW     18$                                     ;
           53  00000000G  00  D0 00126 7$:    MOVL    DBG$GL_LOGFAB+8, STS                    ; 1560
           52  00000000G  00  D0 0012D        MOVL    DBG$GL_LOGFAB+12, STV                   ; 1561
               00000000G  00  D5 00134        TSTL    DBG$GL_LOG_BUF                          ; 1562
                          12  13 0013A        BEQL    8$                                      ;
      18   AE  00000000G  00  9B 0013C        MOVZBW  DBG$GL_LOGFAB+52, MSG_DESC              ;
      1C   AE  00000000G  00  D0 00144        MOVL    DBG$GL_LOGFAB+44, MSG_DESC+4            ;
                          10  11 0014C        BRB     9$                                      ;
      18   AE  00000000G  00  9B 0014E 8$:    MOVZBW  DBG$GL_LOGFAB+53, MSG_DESC              ;
      1C   AE  00000000G  00  D0 00156        MOVL    DBG$GL_LOGFAB+48, MSG_DESC+4            ;
           03             59  E8 0015E 9$:    BLBS    OPENED_LOG, 10$                         ; 1564
                        00C3  31 00161        BRW     16$                                     ;
                          56  DD 00164 10$:   PUSHL   TEMP_NAM                                ; 1565
00000000G  00             01  FB 00166        CALLS   #1, DBG$REL_MEMORY                      ;
                          55  DD 0016D        PUSHL   TEMP_FSR                                ;
00000000G  00             01  FB 0016F        CALLS   #1, DBG$REL_MEMORY                      ;
                          5B  DD 00176        PUSHL   TEMP_FSE                                ;
00000000G  00             01  FB 00178        CALLS   #1, DBG$REL_MEMORY                      ;
00000000G  00             54  D0 0017F        MOVL    OLD_NAM_PTR, DBG$GL_LOGNAM              ;
                          54  D4 00186        CLRL    OLD_NAM_PTR                             ;
00000000G  00  00000000G  00  D0 00188        MOVL    DBG$GL_LOGNAM, DBG$GL_LOGFAB+40         ;
00000000G  00             57  D0 00193        MOVL    FNA, DBG$GL_LOGFAB+44                   ;
00000000G  00             6E  90 0019A        MOVB    FNS, DBG$GL_LOGFAB+52                   ;
00000000G  00             03  88 001A1        BISB2   #3, DBG$GL_LOGFAB+7                     ;
00000000G  00             02  8A 001A8        BICB2   #2, DBG$GL_LOGFAB+4                     ;
                     04   AE  9F 001AF        PUSHAB  TYPE_B                                  ;
    0000V  CF             01  FB 001B2        CALLS   #1, SETUP_LOG                           ;
           64             50  E8 001B7        BLBS    ERR, 15$                                ;
               00000000G  00  D5 001BA        TSTL    DBG$GL_LOG_BUF                          ;
                          12  13 001C0        BEQL    11$                                     ;
      10   AE  00000000G  00  9B 001C2        MOVZBW  DBG$GL_LOGFAB+52, MSG_DESC              ;
      14   AE  00000000G  00  D0 001CA        MOVL    DBG$GL_LOGFAB+44, MSG_DESC+4            ;
                          10  11 001D2        BRB     12$                                     ;
      10   AE  00000000G  00  9B 001D4 11$:   MOVZBW  DBG$GL_LOGFAB+53, MSG_DESC              ;
      14   AE  00000000G  00  D0 001DC        MOVL    DBG$GL_LOGFAB+48, MSG_DESC+4            ;
           01        04   AE  D1 001E4 12$:   CMPL    TYPE_B, #1                              ;
                          10  12 001E8        BNEQ    13$                                     ;
           51  00000000G  00  D0 001EA        MOVL    DBG$GL_LOGFAB+8, STS                    ;
           50  00000000G  00  D0 001F1        MOVL    DBG$GL_LOGFAB+12, STV                   ;
                          0E  11 001F8        BRB     14$                                     ;
           51  00000000G  00  D0 001FA 13$:   MOVL    DBG$GL_LOGRAB+8, STS                    ;
           50  00000000G  00  D0 00201        MOVL    DBG$GL_LOGRAB+12, STV                   ;
                          50  DD 00208 14$:   PUSHL   STV                                     ;
                          51  DD 0020A        PUSHL   STS                                     ;
                     18   AE  9F 0020C        PUSHAB  MSG_DESC                                ;
                          01  DD 0020F        PUSHL   #1                                      ;
               000210A0   8F  DD 00211        PUSHL   #135328                                 ;
00000000G  00             05  FB 00217        CALLS   #5, LIB$SIGNAL                          ;
00000000G  00             5A  90 0021E 15$:   MOVB    LOG_TEMP, DBG$GB_DEF_OUT                ;
                          12  11 00225        BRB     17$                                     ; 1564
               00000000G  00  D4 00227 16$:   CLRL    DBG$GL_LOGFAB+44                        ; 1569
```

```
                    00000000G   00   94 0022D             CLRB    DBG$GL_LOGFAB+52                        ; 1570
                    00000000G   00   D4 00233             CLRL    DBG$GL_LOG_BUF                          ; 1571
                                52   DD 00239 17$:        PUSHL   STV                                     ; 1574
                                53   DD 0023B             PUSHL   STS
                                20   AE 9F 0023D          PUSHAB  MSG_DESC
                                01   DD 00240             PUSHL   #1
                            000210F8 8F   DD 00242        PUSHL   #135416
00000000G   00                      05   FB 00248        CALLS   #5, LIB$SIGNAL
                            50 00000000G 00   D0 0024F 18$:  MOVL   DBG$GL_LOGNAM, R0                    ; 1582
                            0E      34   A0   E9 00256    BLBC    52(R0), 19$
00000000G   00                      02   88 0025A        BISB2   #2, DBG$GL_LOGFAB+7                     ; 1585
00000000G   00                      02   8A 00261        BICB2   #2, DBG$GL_LOGFAB+4                     ; 1586
00000000G   00                      01   88 00268 19$:   BISB2   #1, DBG$GL_LOGFAB+7                     ; 1589
                                08   AE 9F 0026F 20$:    PUSHAB  B_TYPE                                  ; 1600
    0000V   CF                      01   FB 00272        CALLS   #1, SETUP_LOG
                            58              50   D0 00277    MOVL    R0, STATUS
                            03              58   E9 0027A    BLBC    STATUS, 21$                         ; 1601
                                          0146   31 0027D    BRW     34$
                                00000000G   00   D5 00280 21$:  TSTL    DBG$GL_LOG_BUF                  ; 1609
                                            12   13 00286    BEQL    22$
                        18  AE 00000000G   00   9B 00288    MOVZBW  DBG$GL_LOGFAB+52, MSG_DESC
                        1C  AE 00000000G   00   D0 00290    MOVL    DBG$GL_LOGFAB+44, MSG_DESC+4
                                            10   11 00298    BRB     23$
                        18  AE 00000000G   00   9B 0029A 22$:  MOVZBW  DBG$GL_LOGFAB+53, MSG_DESC
                        1C  AE 00000000G   00   D0 002A2    MOVL    DBG$GL_LOGFAB+48, MSG_DESC+4
                            01      08   AE   D1 002AA 23$:  CMPL    B_TYPE, #1                          ; 1610
                                            10   12 002AE    BNEQ    24$
                            53 00000000G   00   D0 002B0    MOVL    DBG$GL_LOGFAB+8, STS                ; 1613
                            52 00000000G   00   D0 002B7    MOVL    DBG$GL_LOGFAB+12, STV               ; 1614
                                            0E   11 002BE    BRB     25$                                 ; 1610
                            53 00000000G   00   D0 002C0 24$:  MOVL    DBG$GL_LOGRAB+8, STS                ; 1618
                            52 00000000G   00   D0 002C7    MOVL    DBG$GL_LOGRAB+12, STV               ; 1619
                            03              59   E8 002CE 25$:  BLBS    OPENED_LOG, 26$                 ; 1622
                                          00C3   31 002D1    BRW     32$
                                            56   DD 002D4 26$:  PUSHL   TEMP_NAM                         ; 1623
00000000G   00                      01   FB 002D6        CALLS   #1, DBG$REL_MEMORY
                                    55   DD 002DD        PUSHL   TEMP_FSR
00000000G   00                      01   FB 002DF        CALLS   #1, DBG$REL_MEMORY
                                    5B   DD 002E6        PUSHL   TEMP_FSE
00000000G   00                      01   FB 002E8        CALLS   #1, DBG$REL_MEMORY
00000000G   00                      54   D0 002EF        MOVL    OLD_NAM_PTR, DBG$GL_LOGNAM
                                    54   D4 002F6        CLRL    OLD_NAM_PTR
00000000G   00 00000000G            00   D0 002F8        MOVL    DBG$GL_LOGNAM, DBG$GL_LOGFAB+40
00000000G   00                      57   D0 00303        MOVL    FNA, DBG$GL_LOGFAB+44
00000000G   00                      6E   90 0030A        MOVB    FNS, DBG$GL_LOGFAB+52
00000000G   00                      03   88 00311        BISB2   #3, DBG$GL_LOGFAB+7
00000000G   00                      02   8A 00318        BICB2   #2, DBG$GL_LOGFAB+4
                                0C   AE   9F 0031F        PUSHAB  TYPE_B
    0000V   CF                      01   FB 00322        CALLS   #1, SETUP_LOG
                            64              50   E8 00327    BLBS    ERR, 31$
                                00000000G   00   D5 0032A    TSTL    DBG$GL_LOG_BUF
                                            12   13 00330    BEQL    27$
                        10  AE 00000000G   00   9B 00332    MOVZBW  DBG$GL_LOGFAB+52, MSG_DESC
                        14  AE 00000000G   00   D0 0033A    MOVL    DBG$GL_LOGFAB+44, MSG_DESC+4
                                            10   11 00342    BRB     28$
                        10  AE 00000000G   00   9B 00344 27$:  MOVZBW  DBG$GL_LOGFAB+53, MSG_DESC
                        14  AE 00000000G   00   D0 0034C    MOVL    DBG$GL_LOGFAB+48, MSG_DESC+4
```

```
                01        0C    AE   D1 00354 28$:    CMPL    TYPE_B, #1                                      ;
                                  10   12 00358       BNEQ    29$                                             ;
                51 00000000G      00   D0 0035A       MOVL    DBG$GL_LOGFAB+8, STS                            ;
                50 00000000G      00   D0 00361       MOVL    DBG$GL_LOGFAB+12, STV                           ;
                                  0E   11 00368       BRB     30$                                             ;
                51 00000000G      00   D0 0036A 29$:  MOVL    DBG$GL_LOGRAB+8, STS                            ;
                50 00000000G      00   D0 00371       MOVL    DBG$GL_LOGRAB+12, STV                           ;
                                  50   DD 00378 30$:  PUSHL   STV                                             ;
                                  51   DD 0037A       PUSHL   STS                                             ;
                          18      AE   9F 0037C       PUSHAB  MSG_DESC                                        ;
                                  01   DD 0037F       PUSHL   #1                                              ;
                    000210A0      8F   DD 00381       PUSHL   #135328                                         ;
000000000G      00                  05   FB 00387       CALLS   #5, LIB$SIGNAL                                  ;
000000000G      00                  5A   90 0038E 31$:  MOVB    LOG_TEMP, DBG$GB_DEF_OUT                        ;
                                  19   11 00395       BRB     33$                                             ;  1622
                    00000000G     00   D4 00397 32$:  CLRL    DBG$GL_LOGFAB+44                                ;  1627
                    00000000G     00   94 0039D       CLRB    DBG$GL_LOGFAB+52                                ;  1628
                    00000000G     00   D4 003A3       CLRL    DBG$GL_LOG_BUF                                  ;  1629
000000000G      00                  01   8A 003A9       BICB2   #1, DBG$GL_LOGFAB+7                             ;  1630
                                  52   DD 003B0 33$:  PUSHL   STV                                             ;  1633
                                  53   DD 003B2       PUSHL   STS                                             ;
                          20      AE   9F 003B4       PUSHAB  MSG_DESC                                        ;
                                  01   DD 003B7       PUSHL   #1                                              ;
                    000210A0      8F   DD 003B9       PUSHL   #135328                                         ;
000000000G      00                  05   FB 003BF       CALLS   #5, LIB$SIGNAL                                  ;
000000000G      00                  5A   90 003C6 34$:  MOVB    LOG_TEMP, DBG$GB_DEF_OUT                        ;  1638
                3F                59   E9 003CD       BLBC    OPENED_LOG, 35$                                 ;  1639
                                  54   DD 003D0       PUSHL   OLD_NAM_PTR                                     ;  1645
000000000G      00                  01   FB 003D2       CALLS   #1, DBG$REL_MEMORY                              ;
                    00000000G     00   DD 003D9       PUSHL   DBG$GB_LOGFSE                                   ;  1646
000000000G      00                  01   FB 003DF       CALLS   #1, DBG$REL_MEMORY                              ;
                    00000000G     00   DD 003E6       PUSHL   DBG$GB_LOGFSR                                   ;  1647
000000000G      00                  01   FB 003EC       CALLS   #1, DBG$REL_MEMORY                              ;
000000000G      00                  55   D0 003F3       MOVL    TEMP_FSR, DBG$GB_LOGFSR                         ;  1648
000000000G      00                  5B   D0 003FA       MOVL    TEMP_FSE, DBG$GB_LOGFSE                         ;  1649
                                  6E   95 00401       TSTB    FNS                                             ;  1654
                                  0A   13 00403       BEQL    35$                                             ;
                          FF      A7   9F 00405       PUSHAB  -1(FNA)                                         ;  1656
000000000G      00                  01   FB 00408       CALLS   #1, DBG$REL_MEMORY                              ;
                                      04 0040F 35$:   RET                                                     ;  1658

; Routine Size:  1040 bytes,    Routine Base:  DBG$CODE + 0649
```

```
;1548          1659  1 ROUTINE setup_log (block_type)  =
;1549          1660  1 !++
;1550          1661  1 ! FUNCTIONAL DESCRIPTION:
;1551          1662  1 !       This routine creates and opens the log file.
;1552          1663  1 !
;1553          1664  1 ! FORMAL PARAMETERS:
;1554          1665  1 !       block_type - indicates whether error to report is for
;1555          1666  1 !                    FAB ( = 1) or RAB ( = 2)
;1556          1667  1 !
;1557          1668  1 ! IMPLICIT INPUTS·
;1558          1669  1 !       The name of the log file to be opened
;1559          1670  1 !
;1560          1671  1 ! IMPLICIT OUTPUTS:
;1561          1672  1 !       The resultant LOG file-spec as resolved by RMS
;1562          1673  1 !
;1563          1674  1 ! ROUTINE VALUE:
;1564          1675  1 !       Returns the RMS status code if there's an error, otherwise returns TRUE
;1565          1676  1 !
;1566          1677  1 ! SIDE EFFECTS:
;1567          1678  1 !       The log file is created and a RAB connected to it.
;1568          1679  1 !--
;1569          1680  1
;1570          1681  2         BEGIN
;1571          1682  2
;1572          1683  2         LOCAL
;1573          1684  2                 status;
;1574          1685  2         MAP
;1575          1686  2                 dbg$gl_log_buf : REF VECTOR [,BYTE];
;1576          1687  2
;1577          1688  2
;1578        L 1689  2 %IF dbg_mod2
;1579        U 1690  2 %THEN
;1580        U 1691  2 $fao_tt_out ('setup esa 1 !AD', .dbg$gl_lognam[nam$b_esl], .dbg$gl_lognam[nam$l_esa]);
;1581        U 1692  2 $fao_tt_out ('rsa 1 !AD', .dbg$gl_lognam[nam$b_rsl],.dbg$gl_lognam[nam$l_rsa]);
;1582          1693  2 %FI
;1583          1694  2
;1584          1695  2         ! See if a log file is already open
;1585          1696  2
;1586          1697  2         IF .dbg$gl_logfab [fab$w_ifi] NEQ 0
;1587          1698  2         THEN
;1588          1699  2                 RETURN (TRUE);
;1589          1700  2
;1590          1701  2         .block_type = 1;
;1591          1702  2         status = $CREATE(FAB = dbg$gl_logfab);
;1592          1703  2
;1593          1704  2         IF NOT .status
;1594          1705  2         THEN
;1595          1706  2                 RETURN (.status) ;
;1596          1707  2
;1597        L 1708  2 %IF dbg_mod2
;1598        U 1709  2 %THEN
;1599        U 1710  2 $fao_tt_out(' about to connect');
;1600          1711  2 %FI
;1601          1712  2         dbg$gl_lograb [rab$l_fab] = dbg$gl_logfab;
;1602          1713  2         .block_type = 2;
;1603          1714  2         status = $CONNECT (RAB = dbg$gl_lograb);
;1604          1715  2
```

```
;  1605        L 1716  2 %IF dbg_mod2
;  1606        U 1717  2 %THEN
;  1607        U 1718  2 $fao_tt_out(' after connect, status : !XL' , .status);
;  1608        U 1719  2 $fao_tt_out('block_type !XL' , .block_type);
;  1609          1720  2 %FI
;  1610          1721  2         IF NOT .status
;  1611          1722  2         THEN
;  1612          1723  2                 RETURN (.status) ;
;  1613          1724  2
;  1614          1725  2
;  1615          1726  2         ! Eventually put out the title line here
;  1616          1727  3         RETURN (TRUE)
;  1617          1728  1         END;


                                                        .EXTRN  SYS$CREATE, SYS$CONNECT

                                        0004 00000 SETUP_LOG:
                                                        .WORD   Save R2                                         ; 1659
                              52 00000000G   00   9E 00002      MOVAB   DBG$GL_LOGFAB, R2
                                         02   A2   B5 00009      TSTW    DBG$GL_LOGFAB+2                         ; 1697
                                              2B   12 0000C      BNEQ    1$
                                    04   BC        01   D0 0000E      MOVL    #1, @BLOCK_TYPE                     ; 1701
                                              52   DD 00012      PUSHL   R2                                      ; 1702
                         00000000G    00        01   FB 00014      CALLS   #1, SYS$CREATE
                                              1E   50   E9 0001B      BLBC    STATUS, 2$                          ; 1704
                         00000000G    00        62   9E 0001E      MOVAB   DBG$GL_LOGFAB, DBG$GL_LOGRAB+60         ; 1712
                                    04   BC        02   D0 00025      MOVL    #2, @BLOCK_TYPE                     ; 1713
                                         00000000G   00   9F 00029      PUSHAB  DBG$GL_LOGRAB                       ; 1714
                         00000000G    00        01   FB 0002F      CALLS   #1, SYS$CONNECT
                                              03   50   E9 00036      BLBC    STATUS, 2$                          ; 1721
                                              50   01   D0 00039 1$:      MOVL    #1, R0                          ; 1727
                                                   04 0003C 2$:      RET                                         ; 1728

; Routine Size:  61 bytes,    Routine Base:  DBG$CODE + 0A59
```

```
; 1619          1729  1 GLOBAL ROUTINE dbg$verify_out (parse_stg_desc) : NOVALUE =
; 1620          1730  1 !++
; 1621          1731  1 ! Functional Description:
; 1622          1732  1 !       Routine figures out whether or not to verify input commands by
; 1623          1733  1 ! consulting the command input stream. It always updates dbg$gb_verptr, even
; 1624          1734  1 ! if VERIFY is not set. This is so we can support an imbedded SET OUTPUT VERIFY
; 1625          1735  1 ! in a command sequence
; 1626          1736  1 !
; 1627          1737  1 ! Formal Parameters:
; 1628          1738  1 !       parse_stg_desc -- The input buffer from which we are taking commands
; 1629          1739  1 !
; 1630          1740  1 ! Implicit Inputs:
; 1631          1741  1 !       dbg$gl_cishead : The head of the command input stream
; 1632          1742  1 !       dbg$gb_verptr  : The pointer to the beginning of the command to
; 1633          1743  1 !                        be verified
; 1634          1744  1 !
; 1635          1745  1 ! Outputs:
; 1636          1746  1 !       None.
; 1637          1747  1 !
; 1638          1748  1 ! Routine Value:
; 1639          1749  1 !       None.
; 1640          1750  1 !
; 1641          1751  1 ! Side Effects:
; 1642          1752  1 !       An input command is "VERIFYed" (copied) to the terminal and/or log file
; 1643          1753  1 !--
; 1644          1754  1
; 1645          1755  2 BEGIN
; 1646          1756  2
; 1647          1757  2
; 1648          1758  2 MAP
; 1649          1759  2         parse_stg_desc  : REF BLOCK [,BYTE];
; 1650          1760  2
; 1651          1761  2 LOCAL
; 1652          1762  2         prev_link       : REF cis$link,
; 1653          1763  2         ok_to_verify,
; 1654          1764  2         buf_len;
; 1655          1765  2
; 1656          1766  2         ! get address of previous link in cis
; 1657          1767  2
; 1658          1768  2         prev_link = .dbg$gl_cishead [cis$a_next_link];
; 1659          1769  2
; 1660          1770  2         CASE .dbg$gl_cishead[cis$b_input_type] FROM cis_dbg$input TO cis_acbuf OF
; 1661          1771  2         SET
; 1662          1772  2
; 1663          1773  2         [cis_dbg$input]:        ok_to_verify = FALSE;
; 1664          1774  2
; 1665          1775  2         [cis_rab]:              IF .prev_link [cis$b_input_type] NEQ cis_inpbuf
; 1666          1776  2                                 THEN
; 1667          1777  2                                         ok_to_verify = TRUE
; 1668          1778  2                                 ELSE
; 1669          1779  3                                         BEGIN
; 1670          1780  3                                         LOCAL pre_prev : REF cis$link;
; 1671          1781  3                                         pre_prev = .prev_link [cis$a_next_link];
; 1672          1782  3                                         IF .pre_prev[cis$b_input_type] NEQ cis_dbg$input
; 1673          1783  3                                         THEN
; 1674          1784  3                                                 ok_to_verify = TRUE
; 1675          1785  3                                         ELSE
```

```
;  1676          1786  3                                       ok_to_verify = FALSE;
;  1677          1787  2                                   END;
;  1678          1788  2
;  1679          1789  2       [cis_inpbuf]:           IF .prev_link [cis$b_input_type] EQL cis_dbg$input
;  1680          1790  2                               THEN
;  1681          1791  2                                   ok_to_verify = FALSE
;  1682          1792  2                               ELSE
;  1683          1793  2                                   ok_to_verify = TRUE;
;  1684          1794  2
;  1685          1795  2       [cis_acbuf]:            ok_to_verify = TRUE;
;  1686          1796  2
;  1687          1797  2       TES;
;  1688          1798  2
;  1689          1799  2       ! buf_len is the number of characters for VERIFY to print from the
;  1690          1800  2       ! command input buffer
;  1691          1801  2
;  1692          1802  2       buf_len = .parse_stg_desc[dsc$a_pointer] - .dbg$gb_verptr;
;  1693          1803  2
;  1694          1804  2       ! Exclamation point (!) is mapped as end-of-line by the lexical
;  1695          1805  2       ! analysers. In order to capture comments for inclusion in VERIFY
;  1696          1806  2       ! reporting we must check each EOL for "!".
;  1697          1807  2
;  1698          1808  2
;  1699          1809  2       IF .dbg$gb_verptr [.buf_len - 1] EQL %C'!'
;  1700          1810  2       AND (.parse_stg_desc [dsc$w_length] AND %X'8000') EQL 0
;  1701          1811  2       THEN
;  1702          1812  2               buf_len = .buf_len + .parse_stg_desc [dsc$w_length] ;
;  1703          1813  2
;  1704          1814  2       IF  .dbg$gb_def_out [out_verify]
;  1705          1815  2       AND .buf_len GTR 1
;  1706          1816  2       AND .ok_to_verify
;  1707          1817  2       THEN
;  1708          1818  2               $fao_tt_out (' !AD',.buf_len, .dbg$gb_verptr);
;  1709          1819  2
;  1710          1820  2       dbg$gb_verptr =  .parse_stg_desc [dsc$a_pointer];
;  1711          1821  2       RETURN
;  1712          1822  1       END;


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

                                        04  00288 P.ACB:   .BYTE   4                                                   ;
                                44  41  21  20  00289       .ASCII  \ !AD\                                              ;


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

                                    003C 00000          .ENTRY  DBG$VERIFY_OUT, Save R2,R3,R4,R5                    ; 1729
                            55 00000000G   00   9E 00002    MOVAB   DBG$GB_VERPTR, R5                                   ;
                            51 00000000G   00   D0 00009    MOVL    DBG$GL_CISHEAD, R1                                  ; 1768
                            50        08   A1   D0 00010    MOVL    8(R1), PREV_LINK                                    ;
                    03      00        02   A1   8F 00014    CASEB   2(R1), #0, #3                                       ; 1770
        0024        0018        000A        0020    00019 1$:   .WORD   4$-1$,-                                         ;
                                                                2$-1$,-                                                 ;
                                                                3$-1$,-                                                 ;
```

```
                                                                5$-1$                                           ;
                          16  11 00021           BRB      4$                                              ; 1773
        02          02    A0  91 00023 2$:       CMPB     2(PREV_LINK), #2                                ; 1775
                          14  12 00027           BNEQ     5$                                              ;
        50          08    A0  D0 00029           MOVL     8(PREV_LINK), PRE_PREV                          ; 1781
                    02    A0  95 0002D           TSTB     2(PRE_PREV)                                     ; 1782
                          07  13 00030           BEQL     4$                                              ;
                          09  11 00032           BRB      5$                                              ; 1784
                    02    A0  95 00034 3$:       TSTB     2(PREV_LINK)                                    ; 1789
                          04  12 00037           BNEQ     5$                                              ;
                          53  D4 00039 4$:       CLRL     OK_TO_VERIFY                                    ; 1791
                          03  11 0003B           BRB      6$                                              ;
        53                01  D0 0003D 5$:       MOVL     #1, OK_TO_VERIFY                                ; 1795
        52          04    AC  D0 00040 6$:       MOVL     PARSE_STG_DESC, R2                              ; 1802
        50                65  D0 00044           MOVL     DBG$GB_VERPTR, R0                               ;
  51        04    A2      50  C3 00047           SUBL3    R0, 4(R2), BUF_LEN                              ;
        21          FF  A140  91 0004C           CMPB     -1(BUF_LEN)[R0], #33                            ; 1809
                          0A  12 00051           BNEQ     7$                                              ;
                          62  B5 00053           TSTW     (R2)                                            ; 1810
                          06  19 00055           BLSS     7$                                              ;
        54                62  3C 00057           MOVZWL   (R2), R4                                        ; 1812
        51                54  C0 0005A           ADDL2    R4, BUF_LEN                                     ;
        19 00000000G      00  E9 0005D 7$:       BLBC     DBG$GB_DEF_OUT+2, 8$                            ; 1814
        01                51  D1 00064           CMPL     BUF_LEN, #1                                     ; 1815
                          14  15 00067           BLEQ     8$                                              ;
        11                53  E9 00069           BLBC     OK_TO_VERIFY, 8$                                ; 1816
                          50  DD 0006C           PUSHL    R0                                              ; 1818
                          51  DD 0006E           PUSHL    BUF_LEN                                         ;
              00000000'   EF  9F 00070           PUSHAB   P.ACB                                           ;
   00000000G  00          03  FB 00076           CALLS    #3, DBG$FAO_OUT                                 ;
        65          04    A2  D0 0007D 8$:       MOVL     4(R2), DBG$GB_VERPTR                            ; 1820
                              04 00081           RET                                                      ; 1822

; Routine Size:  130 bytes,    Routine Base:  DBG$CODE + 0A96


; 1713          1823  1
; 1714          1824  1 END
; 1715          1825  0 ELUDOM



                                                 .EXTRN  LIB$SIGNAL

;                             PSECT SUMMARY
;
;      Name                        Bytes                          Attributes
;
;  DBG$PLIT                          653  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                         2840  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        43         0        1000       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         0         0           7       00:00.2
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1     1545        57         3          97       00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418         1         0          31       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386         1         0          22       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1      150        36        24          12       00:00.3
```

```
;                                  COMMAND QUALIFIERS

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

; Size:         2840 code + 653 data bytes
; Run Time:        00:52.5
; Elapsed Time:    02:42.0
; Lines/CPU Min:    2085
; Lexemes/CPU-Min: 18636
; Memory Used:  275 pages
; Compilation Complete
```

0086 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGMSG
LIS

DBGNCANCL
LIS

DBGNCNTRL
LIS

DBGMOD
LIS