F 1

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

I 8

FILEID**DBGMOD

```
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
```

```
0 MODULE DBGMOD (IDENT = 'V04-000') =
1 BEGIN
1
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 ! FACILITY:     DEBUG
1 !
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 ! ABSTRACT:
1 !
1 ! Version:      1.08
1 !
1 ! History:
1 !       Author:
1 !               Carol Peters, 03 Jul 1976: Version 01
1 !
1 !       Modified by:
1 !               Mike Candela, 18 October 1979
1 !               Richard Title, 12 AUG 1981
1 !
1 ! Revision history:
1 ! 3.00  21-Aug-81         RT      Added support for SET STEP SOURCE and
1 !                                 SET STEP NOSOURCE
1 ! 3.01  20-Oct-81         RT      Added support for SEARCH command.
1 ! 3.02  06-May-82         RT      Added support for SET DEFINE and SHOW DEFINE
1 !       17-Aug-83         PS      Added mode G/NOG_FLOAT
1 !--
```

```
! TABLE OF CONTENTS:
!
FORWARD ROUTINE
        dbg$init_modes:  NOVALUE,                ! SETS MODES TO A SPECIFIED level
        dbg$set_mod_lvl: NOVALUE,
        dbg$set_mod_def: NOVALUE,                ! SETS DEFAULT MODES TO INITIALIZATION values
        dbg$show_mode:   NOVALUE,                ! ACTION ROUTINE TO SHOW DEFAULT modes
        dbg$init_step:   NOVALUE,                ! SETS step LEVELS AS SPECIFIED
        dbg$set_stp_def: NOVALUE,                ! RESETS step LEVELS TO INITIALIZATION VALUES
        dbg$set_stp_lvl: NOVALUE,                ! SETS level OF step
        dbg$show_step:   NOVALUE,                ! ACTION ROUTINE TO SHOW step
        dbg$init_search: NOVALUE,                ! resets search levels to initial
        dbg$set_search_def: NOVALUE,             ! sets search levels as specified
        dbg$set_search_lvl: NOVALUE,             ! set level of search
        dbg$show_search: NOVALUE,                ! action routine for show search
        dbg$init_define: NOVALUE,                ! resets define levels to initial
        dbg$set_define_def: NOVALUE,             ! sets define levels as specified
        dbg$set_define_lvl: NOVALUE,             ! set level of define
        dbg$show_define: NOVALUE,                ! action routine for show define
        dbg$set_out_def: NOVALUE,                ! Sets default OUTPUT config
        dbg$show_output: NOVALUE,                ! Displays OUTPUT configuration
        dbg$set_log:     NOVALUE,                ! Set a new LOG file name
        setup_log:       ,                       ! Creates & opens log file
        dbg$verify_out:  NOVALUE;                ! Decides whether to VERIFY input
!
! REQUIRE FILES:
!
REQUIRE 'SRC$:DBGPROLOG.REQ';
LIBRARY 'LIB$:DBGGEN.L32';

EXTERNAL ROUTINE
        dbg$fao_out: NOVALUE,
        dbg$get_memory,                          ! allocate block of free storage
        dbg$newline: NOVALUE,                    ! output a line
        dbg$nget_trans_radix,                    ! returns translation of default radix
        DBG$CONV_R_50,                           ! converts from radix 50
        dbg$print: NOVALUE,                      ! put a line in the output buffer
        dbg$rel_memory : NOVALUE,                ! return storage to free pool
        dbg$show_radix: NOVALUE;

EXTERNAL
        DBG$GB_KEYPAD_INPUT: BYTE,
        DBG$GL_SCREEN_MODE,
        DBG$GB_DEF_STP :    BLOCKVECTOR
                            [STEP_LEVELS, EVENT$K_STEPPING_DESC_SIZE]
                            FIELD (EVENT$STEPPING_DESC_FIE[DS),
        DBG$GB_STP_PTR :    REF EVENT$STEPPING_DESCRIPTOR, ! current stepping
        dbg$gb_def_search: VECTOR [, BYTE],      ! Block for SEARCH settings
        dbg$gb_search_ptr: REF VECTOR [, BYTE],  ! pointer to search settings
        dbg$gb_def_define: VECTOR [, BYTE],      ! Block for DEFINE settings
        dbg$gb_define_ptr: REF VECTOR [, BYTE],  ! pointer to define settings
        dbg$gb_def_mod: VECTOR [, BYTE],         ! BLOCK FOR MODES
        dbg$gb_def_out: VECTOR [, BYTE],         ! Vector for output config
        dbg$gl_cishead: REF cis$link,            ! Head of cis
        dbg$gb_verptr : REF VECTOR [,BYTE],      ! Verify pointer
        dbg$gb_mod_ptr: REF VECTOR [, BYTE],     ! pointer TO MODES
        dbg$gl_context : BITVECTOR,              ! CONTEXT WORD
```

```
        dbg$gb_language: BYTE,                  ! contains index to current language
        dbg$gl_lognam : REF $NAM_DECL,          ! NAM block for LOG file
        dbg$gl_logfab : BLOCK [,BYTE],          ! FAB for LOG file
        dbg$gl_lograb : BLOCK [,BYTE],          ! RAB for LOG file
        dbg$gb_logfsr : REF VECTOR [,BYTE],     ! Resultant LOG filespec buffer
        dbg$gb_logfse : REF VECTOR [,BYTE],     ! Expanded LOG filespec buffer
        dbg$gl_log_buf,                         ! pointer to buffer containing
                                                ! filespec from SET LOG cmd.
        DBG$OPCODE_KIND_TABLE: VECTOR[512,WORD],
        DBG$OPCODE_NAME_TABLE: BLOCKVECTOR[,10,BYTE];

LITERAL
        dbg_mod         = 0,                    ! DEBUGGING AID FOR mode
        dbg_mod1        = 0,                    ! DEBUGGING AID FOR step
        dbg_mod2        = 0;                    !     "    "   "   log file

!++
! The following table holds the valid mode settings and values
! relevant to them. Each entry in the table is three bytes long.
! The entry has three fields, each of them one byte
! long.
!
!       --------------------------------
!       ! keyword !  offset ! value !
!       --------------------------------
!
! The keyword field holds the token value that represents the particular
! mode, e.g., byte_token indicates that the mode is BYTE.
! The offset field holds the location of the mode setting in any of
! the mode levels.
! The value field holds the value that represents a particular mode
! to DEBUG in an internal sense.
!
LITERAL
        keyword_field   = 0,
        offset_field    = 1,
        value_field     = 2,
        mode_entry_len  = 3,
        step_entry_len  = 3,
        search_entry_len = 3;

BIND
        deflog_name = UPLIT BYTE(%ASCII 'DEBUG.LOG'),
        deflog_size = %CHARCOUNT(%ASCII 'DEBUG.LOG');

! build_err_desc creates a descriptor that contains the filespec for RMS error
! reporting using the shared message file

MACRO
        build_err_desc (msg_desc) =

        IF .dbg$gl_log_buf NEQ 0
        THEN
                BEGIN
                msg_desc [dsc$w_length]  = .dbg$gl_logfab [fab$b_fns];
                msg_desc [dsc$a_pointer] = .dbg$gl_logfab [fab$l_fna];
                END
```

```
        ELSE
                BEGIN
                msg_desc [dsc$w_length] = .dbg$gl_logfab [fab$b_dns];
                msg_desc [dsc$a_pointer] = .dbg$gl_logfab [fab$l_dna];
                END %,

! signal_rms_err reports RMS errors caused by $CREATE, $OPEN or $CONNECT

        signal_rms_err (block_type) =
                BEGIN
                LOCAL
                        msg_desc : BLOCK [8,BYTE],
                        sts,
                        stv ;

                build_err_desc (msg_desc) ;
                IF block_type EQL 1
                THEN
                        BEGIN
                        sts = .dbg$gl_logfab [fab$l_sts];
                        stv = .dbg$gl_logfab [fab$l_stv];
                        END
                ELSE
                        BEGIN
                        sts = .dbg$gl_lograb [rab$l_sts];
                        stv = .dbg$gl_lograb [rab$l_stv];
                        END;
                SIGNAL (shr$_openout+dbg_fac_code, 1, msg_desc, .sts, .stv);
                END %,

! restore_nam restores an RMS NAM block

        restore_nam =
                BEGIN
                LOCAL
                        err,
                        type_b;

                ! release storage for the aborted log file

                dbg$rel_memory ( .temp_nam);
                dbg$rel_memory ( .temp_fsr);
                dbg$rel_memory ( .temp_fse);
                dbg$gl_lognam = .old_nam_ptr;
                old_nam_ptr = 0;
                dbg$gl_logfab [fab$l_nam] = .dbg$gl_lognam;
                dbg$gl_logfab [fab$l_fna] = .fna;
                dbg$gl_logfab [fab$b_fns] = .fns;

                dbg$gl_logfab [fab$v_nam] = 1;
                dbg$gl_logfab [fab$v_cif] = 1;
                dbg$gl_logfab [fab$v_mxv] = 0;
                err = setup_log(type_b);
                IF NOT .err
                THEN
                        signal_rms_err (.type_b) ;
```

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

```
   1 GLOBAL ROUTINE dbg$init_modes (goal_level, source_level) : NOVALUE =
   1 !++
   1 ! FUNCTIONAL DESCRIPTION:
   1 !       Sets all levels from local level to the goal level specified
   1 !       with the mode settings of the source level.
   1 !
   1 ! FORMAL PARAMETERS:
   1 !       goal_level      - highest level to set
   1 !       source_level    - level from which to obtain mode settings
   1 !
   1 ! IMPLICIT INPUTS:
   1 !       none
   1 !
   1 ! IMPLICIT OUTPUTS:
   1 !       THE mode settings of the local level to the goal level
   1 !       are reset to the mode settings of the source level.
   1 !
   1 ! Routine value:
   1 !       NOVALUE
   1 !
   1 ! SIDE EFFECTS:
   1 !       NONE
   1 !--
   1
   2         BEGIN
   2
   2         LOCAL
   2                 temp_level: REF VECTOR [, BYTE];
   2
L  2 %IF dbg_mod
U  2 %THEN
U  2         $fao_tt_out ('copying level !SL modes to level !SL thru !SL',
U  2                         .source_level, .goal_level, local_mode);
   2 %FI
   2
   2         temp_level = dbg$gb_def_mod [.source_level * mode_lvl_size];
   2         DECR I FROM local_mode TO .goal_level DO
   3                 BEGIN
   3                 dbg$gb_mod_ptr = dbg$gb_def_mod [.I * mode_lvl_size];
   3                 dbg$gb_mod_ptr [mode_radix] = .temp_level [mode_radix];
   3                 dbg$gb_mod_ptr [mode_length] = .temp_level [mode_length];
   3                 dbg$gb_mod_ptr [mode_symbols] = .temp_level [mode_symbols];
   3                 dbg$gb_mod_ptr [mode_g_floats] = .temp_level [mode_g_floats];
   3                 dbg$gb_mod_ptr [mode_instruc] = .temp_level [mode_instruc];
   3                 dbg$gb_mod_ptr [mode_ascii] = .temp_level [mode_ascii];
   3                 dbg$gb_mod_ptr [mode_scope] = .temp_level [mode_scope];
   3                 dbg$gb_mod_ptr [mode_globals] = .temp_level [mode_globals];
   3                 dbg$gb_mod_ptr [mode_immediate] = .temp_level [mode_immediate];
   3                 dbg$gb_mod_ptr [mode_fortran] = .temp_level [mode_fortran];
   2                 END;
   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$GL_CISHEAD, DBG$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
```

```
1 !++
1 ! THE FOLLOWING ROUTINES CONTROL INTERPRETATION AND TYPEOUT MODES.
1 !
1 !          ADDRESS INTERPRETATION AND DATA TYPEOUT ARE CONTROLLED BY FOUR MODE
1 !          SETTINGS. THEY ARE 'DEFAULT_MODE', WHICH IS THE MODE SET BY
1 !          DEBUG INITIALIZATION; 'user_def_mode', WHICH IS THE MODE SET
1 !          BY USER 'SET DEFAULT' COMMANDS; 'override_mode', WHICH IS
1 !          THE MODE SET BY A SINGLE LINE OVERRIDE MODE COMMAND; AND
1 !          'local_mode', WHICH IS THE MODE SET FOR A SINGLE VARIABLE.
1 !          WHICH MODE IS USED DEPENDS ON THE pointer INTO THE BLOCK
1 !          dbg$gb_def_mod.
1 !
1 !          AT INITIALIZATION TIME, THE FOUR MODE SETTINGS ARE INITIALIZED
1 !          TO THE SAME values. WHEN THESE values ARE CHANGED BY USER COMMANDS,
1 !          SOME PROPAGATION OF MODE values MUST MOVE UP AND DOWN THROUGH THE
1 !          DIFFERENT BLOCKS SO THAT THE ITEMS DISPLAYED OR INTERPRETED WILL
1 !          HAVE VALID AND TIMELY MODE SETTINGS FOR BOTH RADIX AND LENGTH.
1 !
1 !          THE ROUTINES THAT FOLLOW SET THE MODE BLOCK pointerS TO THEIR NEW
1 !          value AND PROPAGATE values AS NECESSARY.
1 !
1 !          THE GENERAL STRATEGY IS AS FOLLOWS:
1 !          DEFAULT_MODE SETTINGS ARE NEVER CHANGED
1 !          user_def_mode SETTINGS ARE NEVER CHANGED AS A RESULT OF pointer JIGGLING.
1 !                  THEY ARE ONLY CHANGED EXPLICITLY IN ACTION ROUTINES.
1 !          override_mode SETTINGS ARE CHANGED TO REFLECT USER-SET DEFAULTS
1 !                  WHEN THEY ARE THE OBJECT OF THE pointer.
1 !          local_mode SETTINGS ARE COPIED FROM override_mode SETTINGS WHEN THEY
1 !                  ARE THE OBJECT OF THE pointer.
1 !          WHEN THE pointer IS MOVING UP THE BLOCK, FROM local_mode TOWARD
1 !                  DEFAULT_MODE, NO values ARE PROPAGATED.
1 !          SETTING OF local_mode AND override_mode TO DEFAULT SETTINGS AT THE END OF
1 !                  A COMMAND LINE IS THE EXPLICIT RESPONSIBILITY OF EOC ACTION
1 !                  ROUTINE.
1 !--
```

```
1 GLOBAL ROUTINE dbg$set_mod_lvl (level) : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION
1 !       SEE DESCRIPTION ON PAGE 2
1 !
1 ! FORMAL PARAMETERS:
1 !       level   - The structure level to set the pointer to.
1 !                 user_def_mode
1 !                 override_mode
1 !                 local_mode
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! Routine value:
1 !       NO value
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !--
1
2       BEGIN
2
2       LOCAL
2               temp_level: REF VECTOR [, BYTE];
2
3       IF (dbg$gb_def_mod[.level * mode_lvl_size] LEQA .dbg$gb_mod_ptr)
3       OR (.level EQL user_def_mode)
2       THEN
3           BEGIN
3           dbg$gb_mod_ptr = dbg$gb_def_mod [.level * mode_lvl_size];
3           RETURN
2           END;
2
2       SELECT .level OF
2
2               SET
2
2               [override_mode]:
3                       BEGIN
3                       temp_level = dbg$gb_def_mod [user_def_mode * mode_lvl_size];
3                       dbg$gb_mod_ptr = dbg$gb_def_mod [override_mode * mode_lvl_size];
2                       END;
2
2               [local_mode]:
3                       BEGIN
3                       temp_level = .dbg$gb_mod_ptr;
3                       dbg$gb_mod_ptr = dbg$gb_def_mod [local_mode * mode_lvl_size];
2                       END;
2
2               TES;
2       dbg$gb_mod_ptr [mode_radix] = .temp_level [mode_radix];
2       dbg$gb_mod_ptr [mode_length] = .temp_level [mode_length];
2       dbg$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
```

```
1 GLOBAL ROUTINE dbg$set_mod_def : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       CANCELS USER SET DEFAULTS FOR MODE SETTINGS.
1 !       RESETS ALL MODE LEVELS TO LANGUAGE DEFAULTS.
1 !       Called from  DBG$INIT_DEBUG as part of the
1 !       once-only initialization phase of DEBUG.
1 !
1 ! FORMAL PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       THE DEFAULT MODES
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! Routine value:
1 !       NONE VALUE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !--
1
2       BEGIN
2
2       !+
2       ! Set up the MODE data structure
2       !-
2
2       dbg$gb_mod_ptr = dbg$gb_def_mod [default_mode * mode_lvl_size];
2
2       dbg$gb_mod_ptr [mode_radix] = dbg$k_default;
2       dbg$gb_mod_ptr [mode_length] = long_length;
2       dbg$gb_mod_ptr [mode_symbols] = TRUE;
2       dbg$gb_mod_ptr [mode_g_floats] = FALSE;
2       dbg$gb_mod_ptr [mode_instruc] = FALSE;
2       dbg$gb_mod_ptr [mode_ascii] = FALSE;
2       dbg$gb_mod_ptr [mode_scope] = TRUE;
2       dbg$gb_mod_ptr [mode_globals] = FALSE;
2       dbg$gb_mod_ptr [mode_immediate] = TRUE;
2       dbg$gb_mod_ptr [mode_fortran] = 0;
2       dbg$init_modes (user_def_mode, default_mode);
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
```

```
1 GLOBAL ROUTINE dbg$show_mode : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       ACTION ROUTINE TO SHOW MODES
1 !
1 ! FORMAL PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       THE POINTER TO THE CURRENT MODE STRUCTURE.
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! Routine value:
1 !       NOVALUE
1 !
1 ! SIDE EFFECTS:
1 !       THE MODES ARE DISPLAYED ON THE OUTPUT DEVICE
1 !--
2     BEGIN
2     dbg$print(UPLIT BYTE(%ASCIC 'modes: '));
2     IF NOT .dbg$gb_mod_ptr[mode_symbols]
2     THEN
2         dbg$print(UPLIT BYTE(%ASCIC 'no'));
2     dbg$print(UPLIT BYTE(%ASCIC 'symbolic, '));
2     IF NOT .dbg$gb_mod_ptr[mode_g_floats]
2     THEN
2         dbg$print(UPLIT BYTE(%ASCIC 'd_float, '))
2     ELSE
2         dbg$print(UPLIT BYTE(%ASCIC 'g_float, '));
2     IF NOT .dbg$gl_screen_mode
2     THEN
2         dbg$print(UPLIT BYTE(%ASCIC 'no'));
2     dbg$print(UPLIT BYTE(%ASCIC 'screen, '));
2     IF NOT .dbg$gb_keypad_input
2     THEN
2         dbg$print(UPLIT BYTE(%ASCIC 'no'));
2     dbg$print(UPLIT BYTE(%ASCIC 'keypad'));
2     dbg$newline();
2     dbg$show_radix(0);
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
```

```
1  GLOBAL ROUTINE DBG$INIT_STEP (GOAL_LEVEL, SOURCE_LEVEL) : NOVALUE =
1  !++
1  ! FUNCTIONAL DESCRIPTION:
1  !       Sets all levels from override level to the goal level specified
1  !       with the step settings of the source level.
1  !
1  !
1  ! INPUTS:
1  !       goal_level      - highest level to set
1  !       source_level    - level from which to obtain step settings
1  !
1  ! IMPLICIT INPUTS:
1  !       none
1  !
1  ! OUTPUTS:
1  !       NONE
1  !
1  ! IMPLICIT OUTPUTS:
1  !       THE step settings of the override level to the goal level
1  !       are reset to the step settings of the source level.
1  !
1  ! Routine value:
1  !       NOVALUE
1  !
1  ! SIDE EFFECTS:
1  !       NONE
1  !--
2     BEGIN
2     LOCAL
2         TEMP_LEVEL :    REF EVENT$STEPPING_DESCRIPTOR;
2
2     TEMP_LEVEL = DBG$GB_DEF_STP [.SOURCE_LEVEL,
2                                  EVENT$A_STEPPING_DESCRIPTOR
2                                 ];
2     DECR I FROM OVERRIDE_STEP TO .GOAL_LEVEL DO
3         BEGIN
3         DBG$GB_STP_PTR = DBG$GB_DEF_STP [.I, EVENT$A_STEPPING_DESCRIPTOR];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_LINE];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSYSTEM];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_OVER];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_SOURCE];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSILENT];
3
3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] =
3             .TEMP_LEVEL [EVENT$B_STEPPING_KIND];
3         DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] =
3             .TEMP_LEVEL [EVENT$L_STEPPING_OP_LIST];
2         END;
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
```

```
1 GLOBAL ROUTINE DBG$SET_STP_DEF : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       CANCELS USER SET DEFAULTS FOR STEP SETTINGS.
1 !       RESETS ALL STEP LEVELS TO SYSTEM DEFAULTS FOR THE CURRENT LANGUAGE.
1 !
1 !
1 ! INPUTS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       THE DEFAULT STEP SETTINGS
1 !
1 ! OUTPUTS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! Routine value:
1 !       NOVALUE
1 !
1 ! SIDE EFFECTS:
1 !       RESETS DEFAULT values FOR STEP.
1 !--
2     BEGIN
2
2
2     ! Set up the step data structure
2     !
2     DBG$GB_STP_PTR = DBG$GB_DEF_STP [DEFAULT_STEP,
2                                      EVENT$A_STEPPING_DESCRIPTOR
2                                     ];
2     IF .DBG$GB_LANGUAGE EQL DBG$K_MACRO
2     THEN
3         BEGIN
3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] = EVENT$K_INS_EVRY;
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] = FALSE;
3         END
2     ELSE
3         BEGIN
3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] = EVENT$K_INS_LINE;
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] = TRUE;
2         END;
2     DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] = TRUE;
2     DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] = TRUE;
2     DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] = TRUE;
2     DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] = TRUE;
2
2     DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] = 0;
2
2     DBG$INIT_STEP (USER_DEF_STEP, DEFAULT_STEP);
1     END;
```

```
                                          0004 00000         .ENTRY  DBG$SET_STP_DEF, Save R2                            ; 0648
                  52 00000000G     00       9E 00002         MOVAB   DBG$GB_STP_PTR, R2                                  ; 0680
                  62 00000000G     00       9E 00009         MOVAB   DBG$GB_DEF_STP, DBG$GB_STP_PTR
                  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
```

```
1 GLOBAL ROUTINE DBG$SET_STP_LVL (LEVEL) : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION
1 !       This routine sets the pointer to the step structure to the specified
1 !       level.  If the level is not user_def_step the step values there are
1 !       copied to override_step level.  If the level is user_def_step the
1 !       pointer is just reset to that level.
1 !
1 ! FORMAL PARAMETERS:
1 !       LEVEL   - The new step structure level.
1 !                 user_def_step
1 !                 override_step
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! Routine value:
1 !       NO value
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !--
2     BEGIN
2
2     LOCAL
2         TEMP_LEVEL: REF EVENT$STEPPING_DESCRIPTOR;
2
2     IF .LEVEL EQL USER_DEF_STEP
2     THEN
2         DBG$GB_STP_PTR = DBG$GB_DEF_STP [USER_DEF_STEP,
2                                         EVENT$A_STEPPING_DESCRIPTOR
2                                         ]
2     ELSE
3         BEGIN
3         TEMP_LEVEL = DBG$GB_DEF_STP [USER_DEF_STEP,
3                                     EVENT$A_STEPPING_DESCRIPTOR
3                                     ];
3         DBG$GB_STP_PTR = DBG$GB_DEF_STP [OVERRIDE_STEP,
3                                         EVENT$A_STEPPING_DESCRIPTOR
3                                         ];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_LINE] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_LINE];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSYSTEM];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_OVER];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_SOURCE];
3         DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT] =
3             .TEMP_LEVEL [EVENT$V_STEPPING_NOSILENT];
3
3         DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] =
3             .TEMP_LEVEL [EVENT$B_STEPPING_KIND];
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
```

```
GLOBAL ROUTINE DBG$SHOW_STEP : NOVALUE =
!++
! FUNCTIONAL DESCRIPTION:
!       ACTION ROUTINE TO DISPLAY THE CURRENT SETTING OF THE
!       "STEP TYPE" DATA STRUCTURE
!
! INPUTS:
!       NONE
!
! IMPLICIT INPUTS:
!       none.
!
! OUTPUTS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! Routine value:
!       NOVALUE
!
! SIDE EFFECTS:
!       THE STEP SETTINGS ARE DISPLAYED ON THE OUTPUT DEVICE
!--
    BEGIN


    ! Setup a UPLIT %ASCIC string.
    !
    MACRO
        $AC (STRING) =
            UPLIT BYTE (%ASCIC STRING)
        %;


    ! Step type...
    !
    DBG$PRINT   ($AC    ('step type: '));


    ! [NO] SYSTEM....
    !
    IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSYSTEM]
    THEN
        DBG$PRINT   ($AC    ('no'));
    DBG$PRINT   ($AC    ('system, '));


    ! [NO] SOURCE....
    !
    IF NOT .DBG$GB_STP_PTR [EVENT$V_STEPPING_SOURCE]
    THEN
        DBG$PRINT   ($AC    ('no'));
    DBG$PRINT   ($AC    ('source, '));


    ! [NO] SILENT....
```

```
!
IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_NOSILENT]
THEN
    DBG$PRINT   ($AC    ('no'));
DBG$PRINT   ($AC    ('silent, '));


! INTO/OVER....
!
IF .DBG$GB_STP_PTR [EVENT$V_STEPPING_OVER]
THEN
    DBG$PRINT   ($AC    ('over'))
ELSE
    DBG$PRINT   ($AC    ('into'));
DBG$PRINT   ($AC    (' routine calls, '));


! Display the instruction type (calls, branches, all, or user
! list.
!
DBG$PRINT   ($AC ('by !AC'),
                SELECTONE .DBG$GB_STP_PTR [EVENT$B_STEPPING_KIND] OF
                    SET
                    [EVENT$K_INS_CALL] :
                        $AC ('calls:');
                    [EVENT$K_INS_BRAN] :
                        $AC ('branches:');
                    [EVENT$K_EXC_EXC] :
                        $AC ('exceptions');
                    [EVENT$K_INS_EVRY] :
                        $AC ('instruction');
                    [EVENT$K_INS_USER] :
                        $AC ('instruction(s):');
                    [EVENT$K_INS_LINE] :
                        $AC ('line');
                    [EVENT$K_ACC_RTRN] :
                        $AC ('return');
                    TES
            );


! If this entry is an instruction type (/CALL, /BRANCH, etc.) and
! has an opcode bit map (i.e., is not for all instructions),
! display the opcode mnemonics indicated.

IF .DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST] NEQA 0
THEN
    BEGIN
    LOCAL
        OPCODE_LIST :   REF BITVECTOR [512],
        COLUMN;

    ! Point to the opcode list, and set the column to 0.
    !
    OPCODE_LIST = .DBG$GB_STP_PTR [EVENT$L_STEPPING_OP_LIST];
    COLUMN = -1;
    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                 53 E1 000E4  15$:    BBC     OPCODE, (OPCODE_LIST), 17$                        ;
                    50      01         A2 9E 000E8          MOVAB   1(R2), R0                                         ; 0879
   52       50      03                 00 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 0011C          RET                                                       ; 0891
```

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

```
1 GLOBAL ROUTINE dbg$init_search (goal_level, source_level) : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       Sets all levels from override level to the goal level specified
1 !       with the search settings of the source level.
1 !
1 !
1 ! INPUTS:
1 !       goal_level      - highest level to set
1 !       source_level    - level from which to obtain search settings
1 !
1 ! IMPLICIT INPUTS:
1 !       none
1 !
1 ! OUTPUTS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       THE search settings of the override level to the goal level
1 !       are reset to the search settings of the source level.
1 !
1 ! Routine value:
1 !       NOVALUE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !--
1
2       BEGIN
2
2       LOCAL
2               temp_level: REF VECTOR [, BYTE];
2
2       temp_level = dbg$gb_def_search [.source_level * search_lvl_size];
2       DECR I FROM override_search TO .goal_level DO
3               BEGIN
3               dbg$gb_search_ptr = dbg$gb_def_search [.I * search_lvl_size];
3               dbg$gb_search_ptr [search_all] = .temp_level [search_all];
3               dbg$gb_search_ptr [search_ident] = .temp_level [search_ident];
2               END;
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
```

```
GLOBAL ROUTINE dbg$show_search : NOVALUE =
!++
! FUNCTIONAL DESCRIPTION:
!       ACTION ROUTINE TO DISPLAY THE CURRENT SETTING OF THE
!       ''SEARCH TYPE'' DATA STRUCTURE
!
! INPUTS:
!       NONE
!
! IMPLICIT INPUTS:
!       none.
!
! OUTPUTS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! Routine value:
!       NOVALUE
!
! SIDE EFFECTS:
!       THE SEARCH SETTINGS ARE DISPLAYED ON THE OUTPUT DEVICE
!--

BEGIN
        LOCAL
                all_cs          : REF VECTOR[,byte],
                ident_cs        : REF VECTOR[,byte];

        !+
        ! We simply put out one line describing the
        ! current search types, based on the TRUE/FALSE
        ! indicators given in the SEARCH structure.
        !-

        ! If the search mode is not ALL, then it
        ! must be NEXT.

        if( .dbg$gb_search_ptr[ search_all ] )
        then
                all_cs = uplit( %ascic 'all occurences' )
        else
                all_cs = uplit( %ascic 'next occurence' );

        ! If the search mode is not IDENT, then it
        ! must be STRING.

        if( .dbg$gb_search_ptr[ search_ident] )
        then
                ident_cs = uplit( %ascic 'an identifier' )
        else
                ident_cs = uplit( %ascic 'a string');

        ! Put out the line and return.

        $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 43  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$init_define (goal_level, source_level) : NOVALUE =
;  960      1078  1 !++
;  961      1079  1 ! FUNCTIONAL DESCRIPTION:
;  962      1080  1 !       Sets all levels from override level to the goal level specified
;  963      1081  1 !       with the define settings of the source level.
;  964      1082  1 !
;  965      1083  1 ! INPUTS:
;  966      1084  1 !
;  967      1085  1 !       goal_level      - highest level to set
;  968      1086  1 !       source_level    - level from which to obtain define settings
;  969      1087  1 !
;  970      1088  1 ! OUTPUTS:
;  971      1089  1 !
;  972      1090  1 !       NONE
;  973      1091  1 !
;  974      1092  1 ! IMPLICIT OUTPUTS:
;  975      1093  1 !
;  976      1094  1 !       THE define settings of the override level to the goal level
;  977      1095  1 !       are reset to the define settings of the source level.
;  978      1096  1 !
;  979      1097  1 ! Routine value:
;  980      1098  1 !       NOVALUE
;  981      1099  1 !
;  982      1100  1 ! SIDE EFFECTS:
;  983      1101  1 !       NONE
;  984      1102  1 !--
;  985      1103  1
;  986      1104  2       BEGIN
;  987      1105  2
;  988      1106  2       LOCAL
;  989      1107  2           temp_level: REF VECTOR [, BYTE];
;  990      1108  2
;  991      1109  2       temp_level = dbg$gb_def_define [.source_level * define_lvl_size];
;  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
```

```
GLOBAL ROUTINE dbg$show_define : NOVALUE =
!++
! FUNCTIONAL DESCRIPTION:
!       ACTION ROUTINE TO DISPLAY THE CURRENT SETTING OF THE
!       "define TYPE" DATA STRUCTURE
!
! INPUTS:
!       NONE
!
! IMPLICIT INPUTS:
!       none.
!
! OUTPUTS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! Routine value:
!       NOVALUE
!
! SIDE EFFECTS:
!       THE define SETTINGS ARE DISPLAYED ON THE OUTPUT DEVICE
!--

BEGIN

        LOCAL
            def_cs : REF VECTOR [,BYTE];

        !+
        ! We simply put out one line describing the
        ! current define types, based on the values
        ! given in the define structure.
        !-

        CASE .dbg$gb_define_ptr [define_only]
                FROM define_lowest TO define_highest OF
            SET

            [define_address] :
                def_cs = UPLIT (%ASCIC 'ADDRESS');

            [define_command] :
                def_cs = UPLIT (%ASCIC 'COMMAND');

            [define_procedure] :
                def_cs = UPLIT (%ASCIC 'PROCEDURE');

            [define_string] :
                def_cs = UPLIT (%ASCIC 'STRING');

            [define_value] :
                def_cs = UPLIT (%ASCIC 'VALUE');

            [INRANGE, OUTRANGE] :
                $DBG_ERROR('DBGMOD\DBG$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
              06      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
```

```
1 GLOBAL ROUTINE dbg$set_out_def : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       Sets OUTPUT configuration to default settings. Is called from
1 !       DBG$INIT_DEBUG as part of the once only initialization phase.
1 !       Also initializes the default LOG filespec and the string buffers and
1 !       location of the LOG file's NAM block.
1 !
1 ! FORMAL PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !       The default modes
1 !
1 ! IMPLICIT OUTPUTS:
1 !       The vector dbg$gb_def_out is initialized
1 !
1 ! ROUTINE VALUE:
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !       none
1 !--
1
2       BEGIN
2
2
2       dbg$gb_def_out [out_log]        = FALSE;        ! nolog
2       dbg$gb_def_out [out_term]       = TRUE;         ! terminal
2       dbg$gb_def_out [out_verify]     = FALSE;        ! noverify
2
2       ! This variable is used to hold pointer to a user defined filespec
2
2       dbg$gl_log_buf = 0;
2
2       ! initialize default log file spec
2
2       dbg$gl_logfab [fab$l_dna] = deflog_name;
2       dbg$gl_logfab [fab$b_dns] = deflog_size;
2
2       ! Initialize address of related NAM block and the resultant string buffers
2
2       dbg$gl_lognam = dbg$get_memory((nam$c_bln+3)/%UPVAL);
2       $nam_init (nam=.dbg$gl_lognam);
2       dbg$gb_logfse = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
2       dbg$gb_logfsr = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
2       dbg$gl_logfab [fab$l_nam] = .dbg$gl_lognam;
2       dbg$gl_lognam [nam$l_esa] = .dbg$gb_logfse;
2       dbg$gl_lognam [nam$l_rsa] = .dbg$gb_logfsr;
2       dbg$gl_lognam [nam$b_ess] = nam$c_maxrss;
2       dbg$gl_lognam [nam$b_rss] = nam$c_maxrss;
2
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$GB_DEF_OUT                         ;  1293
                             00000000G  00  94 0002E        CLRB    DBG$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        MOVC5   #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
```

```
1 GLOBAL ROUTINE dbg$show_output(full_rep) : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       Action routine to show OUTPUT configuration
1 !
1 ! FORMAL PARAMETERS:
1 !       full_rep    equals 1 for a full report (i.e. "SHOW OUTPUT")
1 !                          2 for just "SHOW LOG"
1 !
1 ! IMPLICIT INPUTS:
1 !       The OUTPUT configuration data structure, dbg$gb_def_out
1 !
1 ! IMPLICIT OUTPUTS:
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !       A message is generated that displays the current output configuration
1 !       and the log file name
1 !--
1
2       BEGIN
2
2       LOCAL
2               verify_cs :     REF VECTOR [,BYTE],
2               term_cs :       REF VECTOR [,BYTE],
2               log_cs :        REF VECTOR [,BYTE],
2               no_count:       word,
2               string_count:   word;                   ! count of chars in output string
2
2       no_count = 0;                                   ! count of "no" characters
2
2       IF .full_rep
2       THEN
3               BEGIN
3               IF .dbg$gb_def_out [out_verify]
3               THEN
3                       verify_cs = UPLIT(%ASCIC '')
3               ELSE
4                       BEGIN
4                       verify_cs = UPLIT(%ASCIC 'no');
4                       no_count = .no_count + 2;
3                       END;
3
3               IF .dbg$gb_def_out [out_term]
3               THEN
3                       term_cs = UPLIT(%ASCIC '')
3               ELSE
4                       BEGIN
4                       term_cs = UPLIT(%ASCIC 'no');
4                       no_count = .no_count + 2
3                       END;
2               END;
2
2       IF .dbg$gb_def_out [out_log]
```

```
        THEN
                log_cs = UPLIT(%ASCIC '')
        ELSE
                BEGIN
                log_cs = UPLIT(%ASCIC 'not ');
                no_count = .no_count + 3;
                END;

%IF dbg_mod2
%THEN
$fao_tt_out('show_output: rss !SB, rsa !XL',.dbg$gl_lognam[nam$b_rss], .dbg$gl_lognam[nam$l_rsa]);
%FI

        ! If a LOG file is open report the filespec in the NAM block, otherwise
        ! use the default filespec

        IF .dbg$gl_logfab [fab$w_ifi] LEQ 0
        THEN
                IF .full_rep
                THEN
                $fao_tt_out('output: !ACverify, !ACterminal, !AClogging to !AD',
                .verify_cs, .term_cs, .log_cs, deflog_size, deflog_name)
                ELSE
                $fao_tt_out('!AClogging to !AD',.log_cs, deflog_size, deflog_name)

        ELSE
            IF .full_rep
            THEN
            BEGIN
            ! string count is initialized to the no. of characters in the output
            ! string (not counting "no"s and the log filespec).

            string_count = 38;
            string_count = .string_count + .no_count + .dbg$gl_lognam[nam$b_rsl];

            ! If we exceed the max record length, output two lines

            IF .string_count LEQ tty_out_width
            THEN
            $fao_tt_out('output: !ACverify, !ACterminal, !AClogging to !AD',
            .verify_cs, .term_cs, .log_cs, .dbg$gl_lognam[nam$b_rsl], .dbg$gl_lognam[nam$l_rsa])

            ELSE
                BEGIN
                $fao_tt_out('output: !ACverify, !ACterminal, !AClogging to :',
                        .verify_cs, .term_cs, .log_cs);
                $fao_tt_out('!AD', .dbg$gl_lognam[nam$b_rsl], .dbg$gl_lognam[nam$l_rsa]);
                END;
            END

            ELSE
            $fao_tt_out('!AClogging to !AD', .log_cs, .dbg$gl_lognam[nam$b_rsl],
                        .dbg$gl_lognam[nam$l_rsa]);

        END;
```

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

                                                     001AE        .BLKB   2
                                 00  00  00  00      001B0 P.ABP: .ASCII  <0><0><0><0>                                      ;
                                 00  6F  6E  02      001B4 P.ABQ: .ASCII  <2>\no\<0>                                        ;
                                 00  00  00  00      001B8 P.ABR: .ASCII  <0><0><0><0>                                      ;
                                 00  6F  6E  02      001BC P.ABS: .ASCII  <2>\no\<0>                                        ;
                                 00  00  00  00      001C0 P.ABT: .ASCII  <0><0><0><0>                                      ;
                 00  00  00  20  74  6F  6E  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
```

```
1 GLOBAL ROUTINE dbg$set_log : NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine closes any log file that is currently open and opens
1 ! the file specified by the user in SET LOG "filespec"
1 !
1 ! FORMAL PARAMETERS:
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !       The RMS-set IFI (internal file identifier field) bit is used to
1 ! determine if a previous log file is still open.
1 !
1 ! The user specified LOG filespec is pointed to by dbg$gl_log_buf
1 !
1 ! IMPLICIT OUTPUTS:
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !       Any previously open log file is closed and a new log file is opened.
1 ! Logging is inhibited for the duration of this routine.
1 !--
1
2       BEGIN
2
2       LOCAL
2               status,
2               opened_log,
2               fns : BYTE,
2               fna,
2               old_nam_ptr,
2               b_type,
2               temp_nam : REF $NAM_DECL,
2               temp_fse : REF VECTOR [,BYTE],
2               temp_fsr : REF VECTOR [,BYTE],
2               log_temp : BYTE;
2
2       opened_log = FALSE ;
2
2       ! inhibit logging until we can redirect the log file
2
2       log_temp = .dbg$gb_def_out [out_log];
2       dbg$gb_def_out [out_log] = FALSE;
2
2       ! see if a log file has been previously opened. If so, close it,
2       ! since it must be CLOSEd befor we can do the new $PARSE.
2
2       IF .dbg$gl_logfab [fab$w_ifi] NEQ 0
2       THEN
3               BEGIN
3
3               opened_log = TRUE;
3               fna = .dbg$gl_logfab [fab$l_fna];
3               fns = .dbg$gl_logfab [fab$b_fns];
```

```
                        status = $CLOSE (FAB = dbg$gl_logfab);
                        IF NOT .status
                        THEN
                                BEGIN
                                LOCAL
                                        msg_desc : BLOCK [8,BYTE];
                                build_err_desc (msg_desc);
                                SIGNAL (shr$_closeout+dbg_fac_code, 1, msg_desc,
                                        .dbg$gl_logfab[fab$l_sts], .dbg$gl_logfab[fab$l_stv]);
                                END;

                        ! get new NAM block, preserve the original

                        temp_nam = dbg$get_memory((nam$c_bln+3)/%UPVAL);
                        $nam_init (nam = .temp_nam);
                        temp_fse = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
                        temp_fsr = dbg$get_memory((nam$c_maxrss+3)/%UPVAL);
                        temp_nam [nam$l_esa] = .temp_fse;
                        temp_nam [nam$l_rsa] = .temp_fsr;
                        temp_nam [nam$b_ess] = nam$c_maxrss;
                        temp_nam [nam$b_rss] = nam$c_maxrss;
                        dbg$gl_logfab [fab$l_nam] = .temp_nam;
                        old_nam_ptr = .dbg$gl_lognam;
                        dbg$gl_lognam = .temp_nam;

                        END;

%IF dbg_mod2
%THEN
$fao_tt_out('dbg$gl_log_buf= !XL',.dbg$gl_log_buf);
%FI

                ! reset the appropriate FAB file process options

                dbg$gl_logfab [fab$v_cif] = 0;
                dbg$gl_logfab [fab$v_mxv] = 1;
                dbg$gl_logfab [fab$v_nam] = 0;

                ! Set up file name, dbg$gl_log_buf contains the pointer to the user
                ! specified filespec.

                IF .dbg$gl_log_buf NEQ 0
                THEN
                        BEGIN

                        MAP
                                dbg$gl_log_buf : REF VECTOR [,BYTE];

                        dbg$gl_logfab [fab$l_fna] = .dbg$gl_log_buf + 1; ! filename addr starts in 2nd byte
                        dbg$gl_logfab [fab$b_fns] = .dbg$gl_log_buf[0];  ! 1st byte is count
%IF dbg_mod2
%THEN
$fao_tt_out('fna !AD',.dbg$gl_logfab[fab$b_fns], .dbg$gl_logfab[fab$l_fna]);
%FI

                        ! Parse the filespec to see if an explicit version number was
```

```
                ! given. If so set the CIF bit, otherwise we must maximize
                ! the version number.

                status = $PARSE (FAB = dbg$gl_logfab);
                IF NOT .status
                THEN
                        BEGIN
                        LOCAL
                                sts,
                                stv,
                                msg_desc : BLOCK [8,BYTE];

                        ! save these because restore_nam will reset them

                        sts = .dbg$gl_logfab [fab$l_sts];
                        stv = .dbg$gl_logfab [fab$l_stv];
                        build_err_desc (msg_desc);

                        IF .opened_log
                        THEN
                                restore_nam
                        ELSE
                                BEGIN
                                dbg$gl_logfab [fab$l_fna] = 0;
                                dbg$gl_logfab [fab$b_fns] = 0;
                                dbg$gl_log_buf  = 0;
                                END;

                        SIGNAL (shr$_syntax + dbg_fac_code, 1, msg_desc, .sts, .stv);
                        END;

%IF dbg_mod2
%THEN
$fao_tt_out('status; !XL', .status);
$fao_tt_out('nam esa- !AD',.dbg$gl_lognam[nam$b_esl], .dbg$gl_lognam[nam$l_esa]);
%FI
                IF .dbg$gl_lognam [nam$v_exp_ver]
                THEN
                        BEGIN
                        dbg$gl_logfab [fab$v_cif] = 1;
                        dbg$gl_logfab [fab$v_mxv] = 0;
                        END;

                dbg$gl_logfab [fab$v_nam] = 1;          ! open by NAM block since
                END;                                    ! we've already parsed the filespec


%IF dbg_mod2
%THEN
$fao_tt_out('fna !AD',.dbg$gl_logfab[fab$b_fns], .dbg$gl_logfab[fab$l_fna]);
%FI

        ! now open the new log file

        status = setup_log(b_type);
        IF NOT .status
        THEN
```

```
                BEGIN
                LOCAL
                        msg_desc : BLOCK [8,BYTE],
                        sts,
                        stv;

                build_err_desc (msg_desc);
                IF .b_type EQL 1
                THEN
                        BEGIN
                        sts = .dbg$gl_logfab [fab$l_sts];
                        stv = .dbg$gl_logfab [fab$l_stv];
                        END
                ELSE
                        BEGIN
                        sts = .dbg$gl_lograb [rab$l_sts];
                        stv = .dbg$gl_lograb [rab$l_stv];
                        END;

                IF .opened_log
                THEN
                        restore_nam
                ELSE
                        BEGIN
                        dbg$gl_logfab [fab$l_fna] = 0;
                        dbg$gl_logfab [fab$b_fns] = 0;
                        dbg$gl_log_buf  = 0;
                        dbg$gl_logfab [fab$v_nam] = 0;
                        END;

                SIGNAL (shr$_openout+dbg_fac_code, 1, msg_desc, .sts, .stv);
                END;

        ! restore logging status and free temp storage

        dbg$gb_def_out [out_log] = .log_temp;
        IF .opened_log
        THEN

                ! Cleanup the "original" NAM block

                BEGIN
                dbg$rel_memory (.old_nam_ptr);
                dbg$rel_memory (.dbg$gb_logfse);
                dbg$rel_memory (.dbg$gb_logfsr);
                dbg$gb_logfsr = .temp_fsr;
                dbg$gb_logfse = .temp_fse;

                ! Release the buffer that holds the log file spec. Since this
                ! is a counted string we have to adjust the address to free

                IF .fns NEQ 0
                THEN
                        dbg$rel_memory(.fna-1);
                END;
        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                                 ;
      00000000G   00             05  FB 00387          CALLS   #5, LIB$SIGNAL                          ;
      00000000G   00             5A  90 0038E 31$:     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
      00000000G   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                                 ;
      00000000G   00             05  FB 003BF          CALLS   #5, LIB$SIGNAL                          ;
      00000000G   00             5A  90 003C6 34$:     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
      00000000G   00             01  FB 003D2          CALLS   #1, DBG$REL_MEMORY                      ;
                      00000000G  00  DD 003D9          PUSHL   DBG$GB_LOGFSE                           ; 1646
      00000000G   00             01  FB 003DF          CALLS   #1, DBG$REL_MEMORY                      ;
                      00000000G  00  DD 003E6          PUSHL   DBG$GB_LOGFSR                           ; 1647
      00000000G   00             01  FB 003EC          CALLS   #1, DBG$REL_MEMORY                      ;
      00000000G   00             55  D0 003F3          MOVL    TEMP_FSR, DBG$GB_LOGFSR                 ; 1648
      00000000G   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
      00000000G   00             01  FB 00408          CALLS   #1, DBG$REL_MEMORY                      ;
                                     04 0040F 35$:     RET                                             ; 1658

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

```
  1 ROUTINE setup_log (block_type) =
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !       This routine creates and opens the log file.
  1 !
  1 ! FORMAL PARAMETERS:
  1 !       block_type - indicates whether error to report is for
  1 !                    FAB ( = 1) or RAB ( = 2)
  1 !
  1 ! IMPLICIT INPUTS:
  1 !       The name of the log file to be opened
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !       The resultant LOG file-spec as resolved by RMS
  1 !
  1 ! ROUTINE VALUE:
  1 !       Returns the RMS status code if there's an error, otherwise returns TRUE
  1 !
  1 ! SIDE EFFECTS:
  1 !       The log file is created and a RAB connected to it.
  1 !--
  1
  2         BEGIN
  2
  2         LOCAL
  2                 status;
  2         MAP
  2                 dbg$gl_log_buf : REF VECTOR [,BYTE];
  2
  2
L 2 %IF dbg_mod2
U 2 %THEN
U 2 $fao_tt_out ('setup esa 1 !AD', .dbg$gl_lognam[nam$b_esl], .dbg$gl_lognam[nam$l_esa]);
U 2 $fao_tt_out ('rsa 1 !AD', .dbg$gl_lognam[nam$b_rsl],.dbg$gl_lognam[nam$l_rsa]);
  2 %FI
  2
  2         ! See if a log file is already open
  2
  2         IF .dbg$gl_logfab [fab$w_ifi] NEQ 0
  2         THEN
  2                 RETURN (TRUE);
  2
  2         .block_type = 1;
  2         status = $CREATE(FAB = dbg$gl_logfab);
  2
  2         IF NOT .status
  2         THEN
  2                 RETURN (.status) ;
  2
L 2 %IF dbg_mod2
U 2 %THEN
U 2 $fao_tt_out(' about to connect');
  2 %FI
  2         dbg$gl_lograb [rab$l_fab] = dbg$gl_logfab;
  2         .block_type = 2;
  2         status = $CONNECT (RAB = dbg$gl_lograb);
  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
```

```
1 GLOBAL ROUTINE dbg$verify_out (parse_stg_desc) : NOVALUE =
1 !++
1 ! Functional Description:
1 !       Routine figures out whether or not to verify input commands by
1 ! consulting the command input stream. It always updates dbg$gb_verptr, even
1 ! if VERIFY is not set. This is so we can support an imbedded SET OUTPUT VERIFY
1 ! in a command sequence
1 !
1 ! Formal Parameters:
1 !       parse_stg_desc -- The input buffer from which we are taking commands
1 !
1 ! Implicit Inputs:
1 !       dbg$gl_cishead : The head of the command input stream
1 !       dbg$gb_verptr  : The pointer to the beginning of the command to
1 !                        be verified
1 !
1 ! Outputs:
1 !       None.
1 !
1 ! Routine Value:
1 !       None.
1 !
1 ! Side Effects:
1 !       An input command is "VERIFYed" (copied) to the terminal and/or log file
1 !--
1
2 BEGIN
2
2
2 MAP
2       parse_stg_desc  : REF BLOCK [,BYTE];
2
2 LOCAL
2       prev_link       : REF cis$link,
2       ok_to_verify,
2       buf_len;
2
2       ! get address of previous link in cis
2
2       prev_link = .dbg$gl_cishead [cis$a_next_link];
2
2       CASE .dbg$gl_cishead[cis$b_input_type] FROM cis_dbg$input TO cis_acbuf OF
2       SET
2
2       [cis_dbg$input]:        ok_to_verify = FALSE;
2
2       [cis_rab]:              IF .prev_link [cis$b_input_type] NEQ cis_inpbuf
2                               THEN
2                                       ok_to_verify = TRUE
2                               ELSE
3                                       BEGIN
3                                       LOCAL pre_prev : REF cis$link;
3                                       pre_prev = .prev_link [cis$a_next_link];
3                                       IF .pre_prev[cis$b_input_type] NEQ cis_dbg$input
3                                       THEN
3                                               ok_to_verify = TRUE
3                                       ELSE
```

```
                                                    ok_to_verify = FALSE;
                                            END;

            [cis_inpbuf]:               IF .prev_link [cis$b_input_type] EQL cis_dbg$input
                                        THEN
                                            ok_to_verify = FALSE
                                        ELSE
                                            ok_to_verify = TRUE;

            [cis_acbuf]:                ok_to_verify = TRUE;

            TES;

            ! buf_len is the number of characters for VERIFY to print from the
            ! command input buffer

            buf_len = .parse_stg_desc[dsc$a_pointer] - .dbg$gb_verptr;

            ! Exclamation point (!) is mapped as end-of-line by the lexical
            ! analysers. In order to capture comments for inclusion in VERIFY
            ! reporting we must check each EOL for "!".


            IF .dbg$gb_verptr [.buf_len - 1] EQL %C'!'
            AND (.parse_stg_desc [dsc$w_length] AND %X'8000') EQL 0
            THEN
                    buf_len = .buf_len + .parse_stg_desc [dsc$w_length] ;

            IF  .dbg$gb_def_out [out_verify]
            AND .buf_len GTR 1
            AND .ok_to_verify
            THEN
                    $fao_tt_out (' !AD',.buf_len, .dbg$gb_verptr);

            dbg$gb_verptr =  .parse_stg_desc [dsc$a_pointer];
            RETURN
            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$GL_CISHEAD, R1                     ; 1768
                50         08  A1  D0  00010               MOVL    8(R1), PREV_LINK
            03  00         02  A1  8F  00014               CASEB   2(R1), #0, #3                          ; 1770
0024        001B          000A   0020  00019 1$:           .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