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
```

FILEID**DBGNEXCTE

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC  TTTTTTTTTT  EEEEEEEEEE
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC  TTTTTTTTTT  EEEEEEEEEE
DD      DD  BB      BB  GG          NN      NN  EE          XX      XX  CC              TT      EE
DD      DD  BB      BB  GG          NN      NN  EE          XX      XX  CC              TT      EE
DD      DD  BB      BB  GG          NNNN    NN  EE            XX  XX    CC              TT      EE
DD      DD  BB      BB  GG          NNNN    NN  EE            XX  XX    CC              TT      EE
DD      DD  BBBBBBBB    GG          NN  NN  NN  EEEEEEEE        XX      CC              TT      EEEEEEEE
DD      DD  BBBBBBBB    GG          NN  NN  NN  EEEEEEEE        XX      CC              TT      EEEEEEEE
DD      DD  BB      BB  GG    GGGGGG  NN    NNNN  EE            XX  XX    CC              TT      EE
DD      DD  BB      BB  GG    GGGGGG  NN    NNNN  EE            XX  XX    CC              TT      EE
DD      DD  BB      BB  GG        GG  NN      NN  EE          XX      XX  CC              TT      EE              ....
DD      DD  BB      BB  GG        GG  NN      NN  EE          XX      XX  CC              TT      EE              ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC      TT      EEEEEEEEEE      ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  EEEEEEEEEE  XX      XX    CCCCCCCC      TT      EEEEEEEEEE      ....

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 DBGNEXCTE (IDENT = 'V04-000') =
0
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1 !
1 ! FACILITY:     DEBUG
1 !
1 ! ABSTRACT:
1 !
1 !       Contained in this module is the routine DBG$NEXECUTE_CMD which uses the
1 !       literal value ot the verb node of the command execution tree to decide
1 !       which command execution network to invoke. In addition to this routine
1 !       which is the highest level command execution network, this module contains
1 !       several routines which are used by more than one command execution network
1 !       during command execution.
1 !
1 ! ENVIRONMENT:  VAX/VMS
1 !
1 ! AUTHOR:       David Plummer, CREATION DATE:   4/15/80
1 !
1 ! VERSION:      V02.2-001
1 !
1 ! MODIFIED BY:
1 !               Richard Title   Sep, 1981       Added support for the TYPE verb.
1 !               RT              Oct, 1981       Added support for the SEARCH verb
1 !               RT              Jan, 1982       Added support for the IF verb
1 !               RT              Jan, 1982       Added support for the WHILE verb
1 !               RT              Jan, 1982       Added support for the REPEAT verb
1 !               RT              Jan, 1982       Added parameters to DBG$NCIS_ADD
1 !               RT              Feb, 1982       Added support for EXITLOOP verb
1 !               RT              Mar, 1982       Added support for DEFINE command
1 !               RT              Apr, 1982       Added support for DECLARE command
1 !               RT              Apr, 1982       Added support for SPAWN command
```

```
;      58          0058  1 !              RT            May, 1982       Added support for ALLOCATE command
;      59          0059  1 !              VJH           Jul, 1982       Added support for SYMBOLIZE command
;      60          0060  1 .              RT            Aug, 1982       Changed DBG$NGET_ADDRESS to check
;      61          0061  1 .                                            for implementation level 3
;      62          0062  1 !              RT            Sep, 1982       Added support for UNDEFINE command
;      63          0063  1 !              PS            Oct, 1982       Added support for CALL command
;      64          0064  1 !              RT            Dec, 1982       Added support for ATTACH command
;      65          0065  1 !              RT            Feb, 1983       Added support for DUMP command
;      66          0066  1 !
;      67          0067  1
;      68          0068  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;      69          0202  1
;      70          0203  1 LIBRARY 'LIB$:DBGGEN.L32';
;      71          0204  1
;      72          0205  1 FORWARD ROUTINE
;      73          0206  1     DBG$NEXECUTE_CMD,                ! Highest level execution network
;      74          0207  1     DBG$NCIS_ADD,                    ! Adds a node to the CIS
;      75          0208  1     DBG$NCIS_OPENICF,                ! Opens an icf node in the CIS
;      76          0209  1     DBG$NCIS_REMOVE,                 ! Removes a node from the CIS
;      77          0210  1     DBG$NGET_ADDRESS;                ! Obtains an Lvalue or Rvalue
;    79          0211   1 EXTERNAL ROUTINE
;    80          0212   1     DBG$DEF_PR_EXIT,                    ! Procedure exit for @ procedures
;    81          0213   1     DBG$DEF_SYM_ADD,                    ! Add defined symbol
;    82          0214   1     DBG$DEF_SYM_FIND,                   ! Look up defined symbol
;    83          0215   1     DBG$DEPOSIT: NOVALUE,               ! Level 3 EXECUTE_DEPOSIT routine
;    84          0216   1     DBG$EVALUATE: NOVALUE,              ! Level 3 EXECUTE_EVALUATE routine
;    85          0217   1     DBG$EXAMINE : NOVALUE,              ! Level 3 EXECUTE_EXAMINE routine
;    86          0218   1     DBG$GET_MEMORY,                     ! Allocate permanent memory
;    87          0219   1     DBG$GET_TEMPMEM,                    ! Allocate temporary memory
;    88          0220   1     DBG$MAKE_VMS_DESC,                  ! Convert Primary Descriptor to
;    89          0221   1                                         !    VMS Descriptor
;    90          0222   1     DBG$NCOPY_DESC,                     ! Copy a descriptor
;    91          0223   1     DBG$NEXECUTE_ALLOCATE,              ! ALLOCATE command execution network
;    92          0224   1     DBG$NEXECUTE_AT_SIGN,               ! @ filespec execution network
;    93          0225   1     DBG$NEXECUTE_ATTACH,                ! ATTACH command execution network
;    94          0226   1     DBG$NEXECUTE_CALL,                  ! CALL command execution network
;    95          0227   1     DBG$NEXECUTE_CANCEL,                ! CANCEL command execution network
;    96          0228   1     DBG$NEXECUTE_DECLARE,               ! DECLARE command execution network
;    97          0229   1     DBG$NEXECUTE_DEFINE,                ! DEFINE command execution network
;    98          0230   1     DBG$NEXECUTE_DELETE,                ! DELETE command execution network
;    99          0231   1     DBG$NEXECUTE_DUMP,                  ! DUMP command execution network
;   100          0232   1     DBG$NEXECUTE_EDIT,                  ! EDIT command execution network
;   101          0233   1     DBG$NEXECUTE_EXIT,                  ! EXIT command execution network
;   102          0234   1     DBG$NEXECUTE_EXITLOOP,              ! EXITLOOP command execution network
;   103          0235   1     DBG$NEXECUTE_FOR,                   ! FOR command execution network
;   104          0236   1     DBG$NEXECUTE_GO,                    ! GO command execution network
;   105          0237   1     DBG$NEXECUTE_HELP,                  ! HELP command execution network
;   106          0238   1     DBG$NEXECUTE_IF,                    ! IF command execution network
;   107          0239   1     DBG$NEXECUTE_REPEAT,                ! REPEAT command execution network
;   108          0240   1     DBG$NEXECUTE_SEARCH,                ! SEARCH command execution network
;   109          0241   1     DBG$NEXECUTE_SET,                   ! SET verb execution network
;   110          0242   1     DBG$NEXECUTE_SHOW,                  ! SHOW verb execution network
;   111          0243   1     DBG$NEXECUTE_SPAWN,                 ! SPAWN verb execution network
;   112          0244   1     DBG$NEXECUTE_STEP,                  ! STEP command execution network
;   113          0245   1     DBG$NEXECUTE_SYMBOLIZE,             ! SYMBOLIZE command execution network
;   114          0246   1     DBG$NEXECUTE_TYPE,                  ! TYPE command execution network
;   115          0247   1     DBG$NEXECUTE_UNDEFINE,              ! UNDEFINE command execution network
;   116          0248   1     DBG$NEXECUTE_WHILE,                 ! WHILE command execution network
;   117          0249   1     DBG$NFREE_DESC,                     ! Release space for a descriptor
;   118          0250   1     DBG$NGET_LVAL,                      ! Obtains a symbol's lvalue from a prim desc
;   119          0251   1     DBG$NGET_SYMID,                     ! Obtain a symid list
;   120          0252   1     DBG$NGET_TYPE,                      ! Obtains a symbol's type form a prim desc
;   121          0253   1     DBG$NMAKE_ARG_VECT,                 ! Constructs a message argument vector
;   122          0254   1     DBG$NOUT_INFO,                      ! Outputs an info message
;   123          0255   1     DBG$REL_MEMORY: NOVALUE,            ! Release permanent memory
;   124          0256   1     DBG$SCR_EXECUTE_DISPLAY_CMD:NOVALUE,! Execute the DISPLAY command
;   125          0257   1     DBG$SCR_EXECUTE_SAVE_CMD: NOVALUE,  ! Execute the SAVE command
;   126          0258   1     DBG$SCR_EXECUTE_SCROLL_CMD: NOVALUE,! Execute the SCROLL command
;   127          0259   1     DBG$SCR_EXECUTE_SELECT_CMD: NOVALUE,! Execute the SELECT command
;   128          0260   1     DBG$STA_LOCK_SYMID: NOVALUE;        ! Lock a SYMID list
;   129          0261   1
;   130          0262   1 EXTERNAL
;   131          0263   1     DBG$GL_CISHEAD: REF CIS$LINK,       ! Version 2 debugger head of command input stream
;   132          0264   1     DBG$GL_CIS_LEVELS,                  ! Count of number of levels of CIS.
;   133          0265   1     DBG$GB_DEF_OUT: VECTOR[,BYTE],      ! Old debugger output vector control
;   134          0266   1     DBG$GL_SCREEN_ERROR,                ! Screen error display pointer (or 0)
;   135          0267   1     DBG$GL_SCREEN_NOGO,                 ! Screen flag to turn off STEP and GO
```

```
        DBG$GL_SCREEN_OUTPUT,                   ! Screen output display pointer (or 0)
        DBG$GL_SCREEN_SOURCE;                   ! Screen source display pointer (or 0)

LITERAL
        ALLOCATE_VERB           = DBG$K_ALLOCATE_VERB,
        AT_SIGN_VERB            = DBG$K_AT_SIGN_VERB,
        ATTACH_VERB             = DBG$K_ATTACH_VERB,
        CALL_VERB               = DBG$K_CALL_VERB,
        CANCEL_VERB             = DBG$K_CANCEL_VERB,
        DECLARE_VERB            = DBG$K_DECLARE_VERB,
        DEFINE_VERB             = DBG$K_DEFINE_VERB,
        DELETE_VERB             = DBG$K_DELETE_VERB,
        DEPOSIT_VERB            = DBG$K_DEPOSIT_VERB,
        DISPLAY_VERB            = DBG$K_DISPLAY_VERB,
        DUMP_VERB               = DBG$K_DUMP_VERB,
        EDIT_VERB               = DBG$K_EDIT_VERB,
        EVALUATE_VERB           = DBG$K_EVALUATE_VERB,
        EXAMINE_VERB            = DBG$K_EXAMINE_VERB,
        EXIT_VERB               = DBG$K_EXIT_VERB,
        EXITLOOP_VERB           = DBG$K_EXITLOOP_VERB,
        FOR_VERB                = DBG$K_FOR_VERB,
        GO_VERB                 = DBG$K_GO_VERB,
        HELP_VERB               = DBG$K_HELP_VERB,
        IF_VERB                 = DBG$K_IF_VERB,
        REPEAT_VERB             = DBG$K_REPEAT_VERB,
        SAVE_VERB               = DBG$K_SAVE_VERB,
        SCROLL_VERB             = DBG$K_SCROLL_VERB,
        SEARCH_VERB             = DBG$K_SEARCH_VERB,
        SELECT_VERB             = DBG$K_SELECT_VERB,
        SET_VERB                = DBG$K_SET_VERB,
        SHOW_VERB               = DBG$K_SHOW_VERB,
        SPAWN_VERB              = DBG$K_SPAWN_VERB,
        STEP_VERB               = DBG$K_STEP_VERB,
        SYMBOLIZE_VERB          = DBG$K_SYMBOLIZE_VERB,
        TYPE_VERB               = DBG$K_TYPE_VERB,
        UNDEFINE_VERB           = DBG$K_UNDEFINE_VERB,
        WHILE_VERB              = DBG$K_WHILE_VERB;


! The following macro verifies entrance to, or exit from an ICF.
!
MACRO
            ICF_MESSAGE (PREFIX) =

            BEGIN
            BIND
                    ENTER_PHRASE = UPLIT BYTE(8, %ASCII 'entering'),
                    EXIT_PHRASE  = UPLIT BYTE(7, %ASCII 'exiting');

            LOCAL
                    PHRASE;

            IF prefix EQL 1
            THEN
                    phrase = enter_phrase
            ELSE
                    phrase = exit_phrase;
```

```
:  193      M 0325  1
:  194      M 0326  1
:  195      M 0327  1       dbg$nout_info (dbg$_verifyicf, 3, .phrase, .fab_ptr[fab$b_fns], .fab_ptr[fab$l_fna]);   ! Info messa
:  196      M 0328  1
:  197        0329  1       END % ;
```

```
1 GLOBAL ROUTINE DBG$NEXECUTE_CMD (VERB_NODE_PTR, MESSAGE_VECT) =
1
1  !
1  ! FUNCTIONAL DESCRIPTION:
1  !       DBG$NEXECUTE_CMD is the highest level command execution network. This
1  !       routine examines the value of the verb node in the command execution
1  !       tree to decide which DEBUG command is to be executed, and transfer to
1  !       an appropriate subnetwork to perform the associated semantic action.
1  !
1  ! FORMAL PARAMETERS:
1  !
1  !       VERB_NODE_PTR -         pointer to the head of the command execution tree
1  !
1  !       MESSAGE_VECT -          address of a longword to contain the address of
1  !                               a message argument vector
1  !
1  ! IMPLICIT INPUTS:
1  !
1  !       NONE
1  !
1  ! IMPLICIT OUTPUTS:
1  !
1  !       NONE
1  !
1  ! ROUTINE VALUE:
1  !
1  !       unsigned integer longword completion code
1
1  ! COMPLETION CODES:
1  !
1  !       STS$K_SEVERE (4) -      The specified command could not be executed
1  !
1  !       STS$K_SUCCESS (1) -     The specified command was executed
1  !
1  ! SIDE EFFECTS:
1  !
1  !       The semantic actions corresponding to the parsed DEBUG command are
1  !       performed. Various states of the debugger and user program may be
1  !       altered, and output may be displayed to the user and written to a log
1  !       file.
1  !
1
2     BEGIN
2
2     LOCAL
2         VERB_NODE : REF DBG$VERB_NODE;          ! Command verb node
2
2
2
2     ! Check for a command to execute.
2     !
2     IF .VERB_NODE_PTR EQL 0 THEN RETURN STS$K_SUCCESS;
2
2
2     ! Obtain the verb node and set the pointer to it to 0.
2     !
2     verb_node = ..verb_node_ptr;
```

```
2     .verb_node_ptr = 0;
2
2
2     ! Now transfer control to the appropriate subnetwork and return
2     !
2     RETURN
3         ( CASE .VERB_NODE [DBG$B_VERB_LITERAL] FROM DBG$K_FIRST_VERB
3                                                         TO DBG$K_LAST_VERB OF
3             SET
3
3             [allocate_verb] :
3                 dbg$nexecute_allocate (.verb_node, .message_vect);
3
3             [at_sign_verb] :
3                 dbg$nexecute_at_sign (.verb_node, .message_vect);
3
3             [attach_verb] :
3                 dbg$nexecute_attach (.verb_node, .message_vect);
3
3             [call_verb] :
3                 dbg$nexecute_call (.verb_node, .message_vect);
3
3             [cancel_verb] :
3                 dbg$nexecute_cancel (.verb_node, .message_vect);
3
3             [declare_verb] :
3                 dbg$nexecute_declare (.verb_node, .message_vect);
3
3             [define_verb] :
3                 dbg$nexecute_define (.verb_node, .message_vect);
3
3             [delete_verb] :
3                 dbg$nexecute_delete (.verb_node, .message_vect);
3
3             [deposit_verb] :
3                 (dbg$deposit(.verb_node);sts$k_success);
3
3             [DISPLAY_VERB]:
4                 (DBG$SCR_EXECUTE_DISPLAY_CMD(.VERB_NODE, FALSE);
3                  STS$K_SUCCESS);
3
3             [dump_verb] :
3                 dbg$nexecute_dump(.verb_node, .message_vect);
3
3             [edit_verb] :
3                 dbg$nexecute_edit(.verb_node, .message_vect);
3
3             [evaluate_verb] :
3                 (dbg$evaluate(.verb_node);sts$k_success);
3
3             [examine_verb] :
3                 (dbg$examine(.verb_node);sts$k_success);
3
3             [exit_verb] :
3                 dbg$nexecute_exit (.verb_node, .message_vect);
3
3             [exitloop_verb] :
```

```
3               dbg$nexecute_exitloop (.verb_node, .message_vect);
3
3           [for_verb] :
3               dbg$nexecute_for (.verb_node, .message_vect);
3
3           [go_verb] :
3               dbg$nexecute_go (.verb_node, .message_vect);
3
3           [help_verb] :
3               dbg$nexecute_help (.verb_node, .message_vect);
3
3           [if_verb] :
3               dbg$nexecute_if (.verb_node, .message_vect);
3
3           [repeat_verb] :
3               dbg$nexecute_repeat (.verb_node, .message_vect);
3
3           [SAVE_VERB]:
4               (DBG$SCR_EXECUTE_SAVE_CMD(.VERB_NODE);
3                STS$K_SUCCESS);
3
3           [SCROLL_VERB]:
4               (DBG$SCR_EXECUTE_SCROLL_CMD(.VERB_NODE);
3                STS$K_SUCCESS);
3
3           [search_verb] :
3               dbg$nexecute_search (.verb_node, .message_vect);
3
3           [SELECT_VERB]:
4               (DBG$SCR_EXECUTE_SELECT_CMD(.VERB_NODE);
3                STS$K_SUCCESS);
3
3           [show_verb] :
3               dbg$nexecute_show (.verb_node, .message_vect);
3
3           [set_verb] :
3               dbg$nexecute_set (.verb_node, .message_vect);
3
3           [spawn_verb] :
3               dbg$nexecute_spawn (.verb_node, .message_vect);
3
3           [step_verb] :
3               dbg$nexecute_step (.verb_node, .message_vect);
3
3           [symbolize_verb] :
3               dbg$nexecute_symbolize (.verb_node, .message_vect);
3
3           [type_verb] :
3               dbg$nexecute_type (.verb_node, .message_vect);
3
3           [undefine_verb] :
3               dbg$nexecute_undefine (.verb_node, .message_vect);
3
3           [while_verb] :
3               dbg$nexecute_while (.verb_node, .message_vect);
3
3           [INRANGE, OUTRANGE] :
```

```
:  370          0501  4                 BEGIN
:  371          0502  4                 .ressage_vect = dbg$nmake_arg_vect (dbg$_notimplan, 1,
:  372          0503  4                                     UPLIT BYTE (17, 'full verb support'));
:  373          0504  4                 sts$k_severe
:  374          0505  3                 END;
:  375          0506  3
:  376          0507  2             TES );
:  377          0508  1         END;


                                                                          .TITLE  DBGNEXCTE
                                                                          .IDENT  \V04-000\

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

                                                         11  00000 P.AAA:  .BYTE   17
6F  70  70  75  73  20  62  72  65  76  20  6C  6C  75  66  00001         .ASCII  \full verb support\
                                                     74  72  00010

                                                                          .EXTRN  DBG$DEF_PR_EXIT
                                                                          .EXTRN  DBG$DEF_SYM_ADD
                                                                          .EXTRN  DBG$DEF_SYM_FIND
                                                                          .EXTRN  DBG$DEPOSIT, DBG$EVALUATE
                                                                          .EXTRN  DBG$EXAMINE, DBG$GET_MEMORY
                                                                          .EXTRN  DBG$GET_TEMPMEM
                                                                          .EXTRN  DBG$MAKE_VMS_DESC
                                                                          .EXTRN  DBG$NCOPY_DESC, DBG$NEXECUTE_ALLOCATE
                                                                          .EXTRN  DBG$NEXECUTE_AT_SIGN
                                                                          .EXTRN  DBG$NEXECUTE_ATTACH
                                                                          .EXTRN  DBG$NEXECUTE_CALL
                                                                          .EXTRN  DBG$NEXECUTE_CANCEL
                                                                          .EXTRN  DBG$NEXECUTE_DECLARE
                                                                          .EXTRN  DBG$NEXECUTE_DEFINE
                                                                          .EXTRN  DBG$NEXECUTE_DELETE
                                                                          .EXTRN  DBG$NEXECUTE_DUMP
                                                                          .EXTRN  DBG$NEXECUTE_EDIT
                                                                          .EXTRN  DBG$NEXECUTE_EXIT
                                                                          .EXTRN  DBG$NEXECUTE_EXITLOOP
                                                                          .EXTRN  DBG$NEXECUTE_FOR
                                                                          .EXTRN  DBG$NEXECUTE_GO
                                                                          .EXTRN  DBG$NEXECUTE_HELP
                                                                          .EXTRN  DBG$NEXECUTE_IF
                                                                          .EXTRN  DBG$NEXECUTE_REPEAT
                                                                          .EXTRN  DBG$NEXECUTE_SEARCH
                                                                          .EXTRN  DBG$NEXECUTE_SET
                                                                          .EXTRN  DBG$NEXECUTE_SHOW
                                                                          .EXTRN  DBG$NEXECUTE_SPAWN
                                                                          .EXTRN  DBG$NEXECUTE_STEP
                                                                          .EXTRN  DBG$NEXECUTE_SYMBOLIZE
                                                                          .EXTRN  DBG$NEXECUTE_TYPE
                                                                          .EXTRN  DBG$NEXECUTE_UNDEFINE
                                                                          .EXTRN  DBG$NEXECUTE_WHILE
                                                                          .EXTRN  DBG$NFREE_DESC, DBG$NGET_LVAL
                                                                          .EXTRN  DBG$NGET_SYMID, DBG$NGET_TYPE
                                                                          .EXTRN  DBG$NMAKE_ARG_VECT
                                                                          .EXTRN  DBG$NOUT_INFO, DBG$REL_MEMORY
                                                                          .EXTRN  DBG$SCR_EXECUTE_DISPLAY_CMD
```

```
                                                                  .EXTRN  DBG$SCR_EXECUTE_SAVE_CMD
                                                                  .EXTRN  DBG$SCR_EXECUTE_SCROLL_CMD
                                                                  .EXTRN  DBG$SCR_EXECUTE_SELECT_CMD
                                                                  .EXTRN  DBG$STA_LOCK_SYMID
                                                                  .EXTRN  DBG$GL_CISHEAD, DBG$GL_CIS_LEVELS
                                                                  .EXTRN  DBG$GB_DEF_OUT, DBG$GL_SCREEN_ERROR
                                                                  .EXTRN  DBG$GL_SCREEN_NOGO
                                                                  .EXTRN  DBG$GL_SCREEN_OUTPUT
                                                                  .EXTRN  DBG$GL_SCREEN_SOURCE

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

                                                   0004 00000            .ENTRY  DBG$NEXECUTE_CMD, Save R2                ; 0330
                                  50       04   AC   D0 00002            MOVL    VERB_NODE_PTR, R0                        ; 0381
                                                03   12 00006            BNEQ    1$                                       ;
                                              019F   31 00008            BRW     30$                                      ;
                                  52            60   D0 0000B 1$:        MOVL    (R0), VERB_NODE                          ; 0386
                                                60   D4 0000E            CLRL    (R0)                                     ; 0387
                       20         01            62   8F 00010            CASEB   (VERB_NODE), #1, #32                     ; 0393
        00AD         0093       0086          006C      00014 2$:        .WORD   4$-2$,-                                  ;
        010F         0104       00F9          00C7      0001C                    6$-2$,-                                  ;
        01C1         019A       01A7          0136      00024                    7$-2$,-                                  ;
        0150         0180       01DB          0143      0002C                    9$-2$,-                                  ;
        00A0         011C       015D          01F5      00034                    11$-2$,-                                 ;
        01E8         01CE       005F          01B4      0003C                    15$-2$,-                                 ;
        00D2         00DF       0079          0129      00044                    16$-2$,-                                 ;
        00BA         016A       018D          0175      0004C                    18$-2$,-                                 ;
                                              00EC      00054                    21$-2$,-                                 ;
                                                                                 32$-2$,-                                 ;
                                                                                 31$-2$,-                                 ;
                                                                                 34$-2$,-                                 ;
                                                                                 22$-2$,-                                 ;
                                                                                 36$-2$,-                                 ;
                                                                                 28$-2$,-                                 ;
                                                                                 23$-2$,-                                 ;
                                                                                 38$-2$,-                                 ;
                                                                                 24$-2$,-                                 ;
                                                                                 19$-2$,-                                 ;
                                                                                 8$-2$,-                                  ;
                                                                                 33$-2$,-                                 ;
                                                                                 3$-2$,-                                  ;
                                                                                 35$-2$,-                                 ;
                                                                                 37$-2$,-                                 ;
                                                                                 20$-2$,-                                 ;
                                                                                 5$-2$,-                                  ;
                                                                                 13$-2$,-                                 ;
                                                                                 12$-2$,-                                 ;
                                                                                 26$-2$,-                                 ;
                                                                                 29$-2$,-                                 ;
                                                                                 25$-2$,-                                 ;
                                                                                 10$-2$,-                                 ;
                                                                                 14$-2$                                   ;
                             00000000'          EF   9F 00056            PUSHAB  P.AAA                                    ; 0503
                                                01   DD 0005C            PUSHL   #1                                       ; 0502
                             00028250           8F   DD 0005E            PUSHL   #164432                                  ;
                  00000000G  00                 03   FB 00064            CALLS   #3, DBG$NMAKE_ARG_VECT                   ;
                         08  BC                 50   D0 0006B            MOVL    R0, @MESSAGE_VECT                        ;
```

```
                           50              04  DD 0006F          MOVL    #4, R0                                  ; 0501
                                               04 00072          RET                                             ;
                                      08   AC  DD 00073 3$:      PUSHL   MESSAGE_VECT                            ; 0398
                                           52  DD 00076          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 00078          CALLS   #2, DBG$NEXECUTE_ALLOCATE               ;
                                               04 0007F          RET                                             ;
                                      08   AC  DD 00080 4$:      PUSHL   MESSAGE_VECT                            ; 0401
                                           52  DD 00083          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 00085          CALLS   #2, DBG$NEXECUTE_AT_SIGN                ;
                                               04 0008C          RET                                             ;
                                      08   AC  DD 0008D 5$:      PUSHL   MESSAGE_VECT                            ; 0404
                                           52  DD 00090          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 00092          CALLS   #2, DBG$NEXECUTE_ATTACH                 ;
                                               04 00099          RET                                             ;
                                      08   AC  DD 0009A 6$:      PUSHL   MESSAGE_VECT                            ; 0407
                                           52  DD 0009D          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 0009F          CALLS   #2, DBG$NEXECUTE_CALL                   ;
                                               04 000A6          RET                                             ;
                                      08   AC  DD 000A7 7$:      PUSHL   MESSAGE_VECT                            ; 0410
                                           52  DD 000AA          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 000AC          CALLS   #2, DBG$NEXECUTE_CANCEL                 ;
                                               04 000B3          RET                                             ;
                                      08   AC  DD 000B4 8$:      PUSHL   MESSAGE_VECT                            ; 0413
                                           52  DD 000B7          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 000B9          CALLS   #2, DBG$NEXECUTE_DECLARE                ;
                                               04 000C0          RET                                             ;
                                      08   AC  DD 000C1 9$:      PUSHL   MESSAGE_VECT                            ; 0416
                                           52  DD 000C4          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 000C6          CALLS   #2, DBG$NEXECUTE_DEFINE                 ;
                                               04 000CD          RET                                             ;
                                      08   AC  DD 000CE 10$:     PUSHL   MESSAGE_VECT                            ; 0419
                                           52  DD 000D1          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 000D3          CALLS   #2, DBG$NEXECUTE_DELETE                 ;
                                               04 000DA          RET                                             ;
                                           52  DD 000DB 11$:     PUSHL   VERB_NODE                               ; 0422
              00000000G   00               01  FB 000DD          CALLS   #1, DBG$DEPOSIT                         ;
                                           3B  11 000E4          BRB     17$                                     ;
                                           7E  D4 000E6 12$:     CLRL    -(SP)                                   ; 0425
                                           52  DD 000E8          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 000EA          CALLS   #2, DBG$SCR_EXECUTE_DISPLAY_CMD         ;
                                           2E  11 000F1          BRB     17$                                     ;
                                      08   AC  DD 000F3 13$:     PUSHL   MESSAGE_VECT                            ; 0429
                                           52  DD 000F6          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 000F8          CALLS   #2, DBG$NEXECUTE_DUMP                   ;
                                               04 000FF          RET                                             ;
                                      08   AC  DD 00100 14$:     PUSHL   MESSAGE_VECT                            ; 0432
                                           52  DD 00103          PUSHL   VERB_NODE                               ;
              00000000G   00               02  FB 00105          CALLS   #2, DBG$NEXECUTE_EDIT                   ;
                                               04 0010C          RET                                             ;
                                           52  DD 0010D 15$:     PUSHL   VERB_NODE                               ; 0435
              00000000G   00               01  FB 0010F          CALLS   #1, DBG$EVALUATE                        ;
                                           7A  11 00116          BRB     27$                                     ;
                                           52  DD 00118 16$:     PUSHL   VERB_NODE                               ; 0438
              00000000G   00               01  FB 0011A          CALLS   #1, DBG$EXAMINE                         ;
                                           6F  11 00121 17$:     BRB     27$                                     ;
                                      08   AC  DD 00123 18$:     PUSHL   MESSAGE_VECT                            ; 0441
                                           52  DD 00126          PUSHL   VERB_NODE                               ;
```

```
         00000000G  00          02  FB 00128         CALLS   #2, DBG$NEXECUTE_EXIT                    ;
                                    04 0012F         RET                                              ;
                          08    AC  DD 00130 19$:    PUSHL   MESSAGE_VECT                             ; 0444
                                52  DD 00133         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 00135         CALLS   #2, DBG$NEXECUTE_EXITLOOP                ;
                                    04 0013C         RET                                              ;
                          08    AC  DD 0013D 20$:    PUSHL   MESSAGE_VECT                             ; 0447
                                52  DD 00140         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 00142         CALLS   #2, DBG$NEXECUTE_FOR                     ;
                                    04 00149         RET                                              ;
                          08    AC  DD 0014A 21$:    PUSHL   MESSAGE_VECT                             ; 0450
                                52  DD 0014D         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 0014F         CALLS   #2, DBG$NEXECUTE_GO                      ;
                                    04 00156         RET                                              ;
                          08    AC  DD 00157 22$:    PUSHL   MESSAGE_VECT                             ; 0453
                                52  DD 0015A         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 0015C         CALLS   #2, DBG$NEXECUTE_HELP                    ;
                                    04 00163         RET                                              ;
                          08    AC  DD 00164 23$:    PUSHL   MESSAGE_VECT                             ; 0456
                                52  DD 00167         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 00169         CALLS   #2, DBG$NEXECUTE_IF                      ;
                                    04 00170         RET                                              ;
                          08    AC  DD 00171 24$:    PUSHL   MESSAGE_VECT                             ; 0459
                                52  DD 00174         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 00176         CALLS   #2, DBG$NEXECUTE_REPEAT                  ;
                                    04 0017D         RET                                              ;
                                52  DD 0017E 25$:    PUSHL   VERB_NODE                                ; 0462
         00000000G  00          01  FB 00180         CALLS   #1, DBG$SCR_EXECUTE_SAVE_CMD             ;
                                21  11 00187         BRB     30$                                      ;
                                52  DD 00189 26$:    PUSHL   VERB_NODE                                ; 0466
         00000000G  00          01  FB 0018B         CALLS   #1, DBG$SCR_EXECUTE_SCROLL_CMD           ;
                                16  11 00192 27$:    BRB     30$                                      ;
                          08    AC  DD 00194 28$:    PUSHL   MESSAGE_VECT                             ; 0470
                                52  DD 00197         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 00199         CALLS   #2, DBG$NEXECUTE_SEARCH                  ;
                                    04 001A0         RET                                              ;
                                52  DD 001A1 29$:    PUSHL   VERB_NODE                                ; 0473
         00000000G  00          01  FB 001A3         CALLS   #1, DBG$SCR_EXECUTE_SELECT_CMD           ;
                    50          01  D0 001AA 30$:    MOVL    #1, R0                                   ;
                                    04 001AD         RET                                              ;
                          08    AC  DD 001AE 31$:    PUSHL   MESSAGE_VECT                             ; 0477
                                52  DD 001B1         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 001B3         CALLS   #2, DBG$NEXECUTE_SHOW                    ;
                                    04 001BA         RET                                              ;
                          08    AC  DD 001BB 32$:    PUSHL   MESSAGE_VECT                             ; 0480
                                52  DD 001BE         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 001C0         CALLS   #2, DBG$NEXECUTE_SET                     ;
                                    04 001C7         RET                                              ;
                          08    AC  DD 001C8 33$:    PUSHL   MESSAGE_VECT                             ; 0483
                                52  DD 001CB         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 001CD         CALLS   #2, DBG$NEXECUTE_SPAWN                   ;
                                    04 001D4         RET                                              ;
                          08    AC  DD 001D5 34$:    PUSHL   MESSAGE_VECT                             ; 0486
                                52  DD 001D8         PUSHL   VERB_NODE                                ;
         00000000G  00          02  FB 001DA         CALLS   #2, DBG$NEXECUTE_STEP                    ;
                                    04 001E1         RET                                              ;
                          08    AC  DD 001E2 35$:    PUSHL   MESSAGE_VECT                             ; 0489
```

```
                                  52  DD 001E5        PUSHL   VERB_NODE                                   ;
            00000000G  0u         02  FB 001E7        CALLS   #2, DBG$NEXECUTE_SYMBOLIZE                  ;
                                  04  001EE           RET                                                 ;
                           08     AC  DD 001EF  36$:  PUSHL   MESSAGE_VECT                                ; 0492
                                  52  DD 001F2        PUSHL   VERB_NODE                                   ;
            00000000G  00         02  FB 001F4        CALLS   #2, DBG$NEXECUTE_TYPE                       ;
                                  04  001FB           RET                                                 ;
                           08     AC  DD 001FC  37$:  PUSHL   MESSAGE_VECT                                ; 0495
                                  52  DD 001FF        PUSHL   VERB_NODE                                   ;
            00000000G  00         02  FB 00201        CALLS   #2, DBG$NEXECUTE_UNDEFINE                   ;
                                  04  00208           RET                                                 ;
                           08     AC  DD 00209  38$:  PUSHL   MESSAGE_VECT                                ; 0498
                                  52  DD 0020C        PUSHL   VERB_NODE                                   ;
            00000000G  00         02  FB 0020E        CALLS   #2, DBG$NEXECUTE_WHILE                      ;
                                  04  00215           RET                                                 ; 0508

; Routine Size:  534 bytes,    Routine Base:  DBG$CODE + 0000
```

```
1 GLOBAL ROUTINE DBG$NCIS_ADD (POINTER, LENGTH, TYPE,
1                                 REPEAT_COUNT, WHILE_CLAUSE, LOOP_INCR) =
1 !
1 ! FUNCTION
1 !       This routine creates and adds a new Command Input Stream (CIS) Entry
1 !       to the Command Input Stream Stack.  The global variable DBG$GL_CISHEAD
1 !       is set to point to the new CIS Entry so that DEBUG commands are gotten
1 !       from this new CIS Entry first.  The forward link in the new entry is
1 !       set to contain the old value of DBG$GL_CISHEAD so that the previous
1 !       CIS entry is restored once the new CIS entry is emptied of commands.
1 !
1 ! INPUTS
1 !       POINTER - The address of either a buffer or a RAB to be placed
1 !                 in the DSC$A_POINTER field of the new link.
1 !
1 !       LENGTH  - The length of the above buffer (0 for RAB).
1 !
1 !       TYPE    - The type of the link to be added.
1 !
1 !       REPEAT_COUNT - The count for a CIS of type CIS_REPEAT.  For a CIS of
1 !                 type FOR, this contains the upper bound.
1 !
1 !       WHILE_CLAUSE - A counted string with the action clause for a CIS of
1 !                 type CIS_WHILE.  For a CIS of type FOR, this contains the
1 !                 name of the loop variable.
1 !
1 !       LOOP_INCR - The loop increment in FOR loops.
1 !
1 !
1 ! OUTPUTS
1 !       This routine returns STS$K_SUCCESS as its value.
1 !
1
2     BEGIN
2
2     MAP
2       WHILE_CLAUSE: REF VECTOR [,BYTE];
2
2     LOCAL
2       FOR_LOOP_VAR,                   ! Points to counted string with FOR
2                                       !      loop variable
2       FOR_UPPER_BOUND,                ! Integer with upper bound for FOR loops
2       TEMP;                           ! Temporary pointer to head CIS node
2
2
2
2     ! Increment the count of the number of levels of CIS we have.
2     !
2     DBG$GL_CIS_LEVELS = .DBG$GL_CIS_LEVELS + 1;
2
2
2     ! Pick up the FOR-loop bounds if this is a FOR-loop CIS.
2     !
2     FOR_LOOP_VAR = .WHILE_CLAUSE;
2     FOR_UPPER_BOUND = .REPEAT_COUNT;
2
2
```

```
;  436          0566  2        ! Save current list head and allocate a new one
;  437          0567  2        !
;  438          0568  2        TEMP = .DBG$GL_CISHEAD ;
;  439          0569  2        DBG$GL_CISHEAD = DBG$GET_MEMORY ((CIS_ELEMENTS+3)/%UPVAL);
;  440          0570  2        DBG$GL_CISHEAD [CIS$A_NEXT_LINK] = .TEMP;
;  441          0571  2        DBG$GL_CISHEAD [CIS$A_INPUT_PTR] = .POINTER;
;  442          0572  2        DBG$GL_CISHEAD [CIS$B_INPUT_TYPE] = .TYPE;
;  443          0573  2        DBG$GL_CISHEAD [CIS$W_LENGTH]       = .LENGTH;
;  444          0574  2
;  445          0575  2        IF .TYPE EQL CIS_REPEAT
;  446          0576  2        THEN
;  447          0577  2            DBG$GL_CISHEAD [CIS$L_REPEAT_COUNT] = .REPEAT_COUNT;
;  448          0578  2
;  449          0579  2        IF .TYPE EQL CIS_WHILE
;  450          0580  2        THEN
;  451          0581  2            DBG$GL_CISHEAD [CIS$V_WHILE_FLAG] = .WHILE_CLAUSE;
;  452          0582  2
;  453          0583  2        IF .TYPE EQL CIS_FOR
;  454          0584  2        THEN
;  455          0585  3            BEGIN
;  456          0586  3            DBG$GL_CISHEAD [CIS$L_FOR_UPPER_BOUND] = .FOR_UPPER_BOUND;
;  457          0587  3            DBG$GL_CISHEAD [CIS$A_FOR_LOOP_VAR] = .FOR_LOOP_VAR;
;  458          0588  3            DBG$GL_CISHEAD [CIS$L_FOR_LOOP_INCR] = .LOOP_INCR;
;  459          0589  2            END;
;  460          0590  2
;  461          0591  2
;  462          0592  2        ! The fields INIT_ADDR and INIT_LENGTH are used to determine
;  463          0593  2        ! how much storage to release for this buffer, since the pointer
;  464          0594  2        ! field is modified by the parser among others.
;  465          0595  2        !
;  466          0596  2        DBG$GL_CISHEAD [CIS$A_INIT_ADDR]    = .POINTER;
;  467          0597  2
;  468          0598  2
;  469          0599  2        ! If we are adding an input buffer add 1 byte to the length
;  470          0600  2        ! to be released because we allocated an extra one so we could
;  471          0601  2        ! guarantee a zero byte at the end of the string.
;  472          0602  2        !
;  473          0603  2        IF .TYPE EQL CIS_INPBUF
;  474          0604  2        THEN
;  475          0605  2            DBG$GL_CISHEAD [CIS$W_INIT_LENGTH]      = .LENGTH + 1
;  476          0606  2
;  477          0607  2        ELSE
;  478          0608  2            DBG$GL_CISHEAD [CIS$W_INIT_LENGTH]      = .LENGTH;
;  479          0609  2
;  480          0610  2        RETURN STS$K_SUCCESS;
;  481          0611  2
;  482          0612  1        END;



                                      003C 00000         .ENTRY  DBG$NCIS_ADD, Save R2,R3,R4,R5                 : 0509
                  55 00000000G    00   9E 00002         MOVAB   DBG$GL_CISHEAD, R5                              :
                     00000000G    00   D6 00009         INCL    DBG$GL_CIS_LEVELS                               : 0557
                  53          10  AC   7D 0000F         MOVQ    REPEAT_COUNT, FOR_UPPER_BOUND                   : 0563
                  52              65   D0 00013         MOVL    DBG$GL_CISHEAD, TEMP                            : 0568
```

```
                                              0E  DD 00016        PUSHL   #14                                 ; 0569
                       00000000G   00         01  FB 00018        CALLS   #1, DBG$GET_MEMORY                  ;
                                   65         50  D0 0001F        MOVL    R0, DBG$GL_CISHEAD                  ;
                              08   A0         52  D0 00022        MOVL    TEMP, 8(R0)                         ; 0570
                              04   A0    04   AC  D0 00026        MOVL    POINTER, 4(R0)                      ; 0571
                                   51    0C   AC  D0 0002B        MOVL    TYPE, R1                            ; 0572
                              02   A0         51  90 0002F        MOVB    R1, 2(R0)                           ;
                                   60    08   AC  B0 00033        MOVW    LENGTH, (R0)                        ; 0573
                                   04         51  D1 00037        CMPL    R1, #4                              ; 0575
                                              05  12 0003A        BNEQ    1$                                  ;
                              18   A0    10   AC  D0 0003C        MOVL    REPEAT_COUNT, 24(R0)                ; 0577
                                   05         51  D1 00041 1$:    CMPL    R1, #5                              ; 0579
                                              07  12 00044        BNEQ    2$                                  ;
     12    A0           01         01    14   AC  F0 00046        INSV    WHILE_CLAUSE, #1, #1, 18(R0)        ; 0581
                                   07         51  D1 0004D 2$:    CMPL    R1, #7                              ; 0583
                                              09  12 00050        BNEQ    3$                                  ;
                              18   A0         53  7D 00052        MOVQ    FOR_UPPER_BOUND, 24(R0)             ; 0586
                              20   A0    18   AC  D0 00056        MOVL    LOOP_INCR, 32(R0)                   ; 0588
                              0C   A0    04   AC  D0 0005B 3$:    MOVL    POINTER, 12(R0)                     ; 0596
                                   02         51  D1 00060        CMPL    R1, #2                              ; 0603
                                              08  12 00063        BNEQ    4$                                  ;
                10    A0      08   AC         01  A1 00065        ADDW3   #1, LENGTH, 16(R0)                  ; 0605
                                              05  11 0006B        BRB     5$                                  ;
                              10   A0    08   AC  B0 0006D 4$:    MOVW    LENGTH, 16(R0)                      ; 0608
                                   50         01  D0 00072 5$:    MOVL    #1, R0                              ; 0610
                                                  04 00075        RET                                         ; 0612

; Routine Size:  118 bytes,    Routine Base:  DBG$CODE + 0216
```

```
1 GLOBAL ROUTINE DBG$NCIS_OPENICF (MESSAGE_VECT) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !       Routine is called when there is a RAB at the top of the command
1 !       input stream. It opens the related FAB and connects the RAB to it
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       message_vect    - address of a longword to contain address of message vector
1 !
1 ! IMPLICIT INPUTS:
1 !       The head of the command input stream
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       on failure, a message argument vector
1 !
1 ! ROUTINE VALUE:
1 !
1 !       sts$k_success (1) - action performed
1 !
1 !       sts$k_severe (4) - failure
1 !
1 ! SIDE EFFECTS:
1 !       A FAB is opened and a RAB connected to it. If SET OUTPUT VERIFY, then
1 !       a message is generated indicating we are entering an indirect command file
1 !--
2     BEGIN
2
2     LOCAL
2         STATUS,                         ! Holds RMS status rode
2         FAB_PTR : REF $FAB_DECL,        ! File access block pointer
2         RAB_PTR : REF $RAB_DECL;        ! Record access block pointer
2
2     ! Extract the related FAB from the RAB at the top of the cis
2     !
2     rab_ptr = .dbg$gl_cishead [cis$a_input_ptr];
2     fab_ptr = .rab_ptr [rab$l_fab];
2
2     status = $OPEN (FAB=.fab_ptr);
2     IF NOT .status
2     THEN
3         BEGIN
3
3         LOCAL
3             MSG_DESC : REF dbg$stg_desc;    ! String descriptor for message
3
3         msg_desc = dbg$get_tempmem (2);
3
3         msg_desc[dsc$w_length]  = .fab_ptr[fab$b_fns];
3         msg_desc[dsc$a_pointer] = .fab_ptr[fab$l_fna];
3
3
3         ! Flag link for removal so we won't try to read from it again
3         !
3         dbg$gl_cishead[cis$v_rem_flag] = 1;
3
```

```
;  3          .message_vect = dbg$nmake_arg_vect (shr$_openin + dbg_fac_code,
;  3                                               1,
;  3                                              .msg_desc, .fab_ptr[fab$l_sts], .fab_ptr[fab$l_stv]);
;  3
;  3          RETURN sts$k_severe;
;  3
;  2          END;
;  2
;  2
;  2      ! Connect the RAB to the just opened FAB
;  2      !
;  2      status = $CONNECT (RAB=.rab_ptr);
;  2      IF NOT .status
;  2      THEN
;  3          BEGIN
;  3          LOCAL
;  3                  MSG_DESC : REF dbg$stg_desc; ! string descriptor for message
;  3
;  3          msg_desc = dbg$get_tempmem (2);
;  3
;  3          msg_desc[dsc$w_length]  = .fab_ptr[fab$b_fns];
;  3          msg_desc[dsc$a_pointer] = .fab_ptr[fab$l_fna]·
;  3
;  3
;  3          ! Flag link for removal so we won't try to read from it again
;  3          !
;  3          dbg$gl_cishead[cis$v_rem_flag] = 1;
;  3
;  3          .message_vect = dbg$nmake_arg_vect (shr$_openin + dbg_fac_code,
;  3                                               1, .msg_desc,
;  3                                               .fab_ptr[fab$l_sts],
;  3                                               .fab_ptr[fab$l_stv]);
;  3
;  3          RETURN sts$k_severe;
;  3
;  2          END;
;  2
;  2
;  2      ! Check for verification message.
;  2      !
;  2      IF .dbg$gb_def_out [out_verify]
;  2      THEN
;  2          icf_message(1);
;  2
;  2      RETURN sts$k_success;
;  2
;  1      END;


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

                              08  00012 P.AAB:     .BYTE    8                                   ;
67  6E  69  72  65  74  6E  65  00013             .ASCII   \entering\                          ;
                              07  0001B P.AAC:     .BYTE    7                                   ;
    67  6E  69  74  69  78  65  0001C             .ASCII   \exiting\                           ;
```

```
                                                ENTER_PHRASE=         P.AAB
                                                EXIT_PHRASE=          P.AAC
                                                      .EXTRN  SYS$OPEN, SYS$CONNECT

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

                           003C 00000                 .ENTRY  DBG$NCIS_OPENICF, Save R2,R3,R4,R5              ; 0613
          55 00000000G  00  9E 00002                  MOVAB   DBG$GL_CISHEAD, R5                              ;
          50            65  D0 00009                  MOVL    DBG$GL_CISHEAD, R0                              ; 0649
          53       04   A0  D0 0000C                  MOVL    4(R0), RAB_PTR                                  ;
          52       3C   A3  D0 00010                  MOVL    60(RAB_PTR), FAB_PTR                            ; 0650
                        52  DD 00014                  PUSHL   FAB_PTR                                         ; 0652
 00000000G  00          01  FB 00016                  CALLS   #1, SYS$OPEN                                    ;
            54          50  D0 0001D                  MOVL    R0, STATUS                                      ;
            0F          54  E9 00020                  BLBC    STATUS, 1$                                      ; 0653
                        53  DD 00023                  PUSHL   RAB_PTR                                         ; 0681
 00000000G  00          01  FB 00025                  CALLS   #1, SYS$CONNECT                                 ;
            54          50  D0 0002C                  MOVL    R0, STATUS                                      ;
            36          54  E8 0002F                  BLBS    STATUS, 2$                                      ; 0682
                        02  DD 00032  1$:             PUSHL   #2                                              ; 0688
 00000000G  00          01  FB 00034                  CALLS   #1, DBG$GET_TEMPMEM                             ;
            60     34   A2  9B 0003B                  MOVZBW  52(FAB_PTR), (MSG_DESC)                         ; 0690
       04   A0     2C   A2  D0 0003F                  MOVL    44(FAB_PTR), 4(MSG_DESC)                        ; 0691
            51          65  D0 00044                  MOVL    DBG$GL_CISHEAD, R1                              ; 0696
       12   A1          01  88 00047                  BISB2   #1, 18(R1)                                      ;
            7E     08   A2  7D 0004B                  MOVQ    8(FAB_PTR), -(SP)                               ; 0700
                        50  DD 0004F                  PUSHL   MSG_DESC                                        ; 0699
                        01  DD 00051                  PUSHL   #1                                              ; 0698
             00021098   8F  DD 00053                  PUSHL   #135320                                         ;
 00000000G  00          05  FB 00059                  CALLS   #5, DBG$NMAKE_ARG_VECT                          ;
       04   BC          50  D0 00060                  MOVL    R0, @MESSAGE_VECT                               ;
            50          04  D0 00064                  MOVL    #4, R0                                          ; 0703
                            04 00067                  RET                                                     ;
          1F 00000000G  00  E9 00068  2$:             BLBC    DBG$GB_DEF_OUT+2, 3$                            ; 0710
          50 00000000'  EF  9E 0006F                  MOVAB   ENTER_PHRASE, PHRASE                            ; 0712
                   2C   A2  DD 00076                  PUSHL   44(FAB_PTR)                                     ;
            7E     34   A2  9A 00079                  MOVZBL  52(FAB_PTR), -(SP)                              ;
                        50  DD 0007D                  PUSHL   PHRASE                                          ;
                        03  DD 0007F                  PUSHL   #3                                              ;
             0002808B   8F  DD 00081                  PUSHL   #163979                                         ;
 00000000G  00          05  FB 00087                  CALLS   #5, DBG$NOUT_INFO                               ;
            50          01  D0 0008E  3$:             MOVL    #1, R0                                          ; 0714
                            04 00091                  RET                                                     ; 0716

; Routine Size:  146 bytes,    Routine Base:  DBG$CODE + 028C


;  588         0717  1
```

```
1 GLOBAL ROUTINE DBG$NCIS_REMOVE(EXIT_FLAG, MESSAGE_VECT) =
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       Removes the top link from the command input stream and delete the
1 !       storage for it. If the link has additional dynamic storage related to
1 !       it, such as a FAB,RAB, input buffer etc., that storage is freed also.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       EXIT_FLAG       - TRUE if this routine is called from EXIT or EXITLOOP.
1 !
1 !       MESSAGE_VECT    - The address of a longword to contain the address of
1 !                         a message argument vector.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The head of the command input stream.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       On error, a message argument vector is constructed and returned.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       STS$K_SUCCESS (1) - Success. Actions performed.
1 !
1 !       STS$K_SEVERE (4) - Failure. Error message argument vector constructed.
1 !
1 ! SIDE EFFECTS:
1 !       The head of the command input stream is reset to what was the
1 !       "next" link before this routine was called. If SET OUTPUT VERIFY,
1 !       then a message is generated saying we are exiting the indirect
1 !       command file.
1 !
1
2     BEGIN
2
2     LOCAL
2         BOUNDS_MATCH,               ! TRUE when FOR loop lower bound matches upper bound
2         BUFLIST: REF VECTOR[],      !
2         COND,                       ! TRUE or FALSE: condition value in WHILE cis
2         DUMMY,                      ! dummy output parameter
2         GLOBAL_FLAG,                ! output param for DEF_SYM_FIND
2         KIND,                       ! kind of define symbol
2         LOOP_INCR,                  ! the loop increment
2         NEW_NAME,                   ! Pointer to the loop variable name
2         NEW_VALPTR: REF DBG$VALDESC,! pointer to a value descriptor
2         SIZE,                       ! Size of loop variable name
2         SYMID_LIST,                 ! list of symids
2         TEMP,                       ! temporary pointer to cis node
2         TYPE,                       ! cis node type
2         VALPTR: REF DBG$VALDESC,    ! pointer to a value descriptor
2         VALUE,                      ! value in value descriptor
2         VARNAME:REF VECTOR[,BYTE],  ! name for FOR loop var
2         WHILE_FLAG;                 ! TRUE for WHILE cis
2
2
```

```
2
2      ! Decrement the count of the number of CIS levels we have.
2      !
2      DBG$GL_CIS_LEVELS = .DBG$GL_CIS_LEVELS - 1;
2
2
2      ! If top link is an input buffer, release the storage for that buffer.
2      !
2      IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_INPBUF
2      THEN
2          DBG$REL_MEMORY(.DBG$GL_CISHEAD[CIS$A_INIT_ADDR]);
2
2
2      ! Also release storage for any other buffers that may have been
2      ! allocated during processing of this line (new buffers get allocated
2      ! when symbols defined by DEFINE/COMMAND are expanded).
2      !
2      BUFLIST = .DBG$GL_CISHEAD[CIS$A_BUFLIST];
2      WHILE .BUFLIST NEQ 0 DO
3          BEGIN
3          DBG$REL_MEMORY(.BUFLIST[1]);
3          TEMP = .BUFLIST[0];
3          DBG$REL_MEMORY(.BUFLIST);
3          BUFLIST = .TEMP;
2          END;
2      DBG$GL_CISHEAD[CIS$A_BUFLIST] = 0;
2
2
2      ! If the top Command Input Steam Entry is a SCREEN CIS Entry, we must reset
2      ! the screen displays to which print, source, and error output are directed
2      ! to be the same as they were before this CIS Entry was added to the Command
2      ! Input Stream.  We also reset the NOGO flag which disables STEP and GO
2      ! commands inside screen display DEBUG command lists.
2      !
2      IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL DBG$K_CIS_SCREEN
2      THEN
3          BEGIN
3          DBG$GL_SCREEN_NOGO = .DBG$GL_CISHEAD[CIS$V_SCREEN_NOGO];
3          DBG$GL_SCREEN_OUTPUT = .DBG$GL_CISHEAD[CIS$L_SCREEN_OUTPUT];
3          DBG$GL_SCREEN_SOURCE = .DBG$GL_CISHEAD[CIS$L_SCREEN_SOURCE];
3          DBG$GL_SCREEN_ERROR  = .DBG$GL_CISHEAD[CIS$L_SCREEN_ERROR];
2          END;
2
2
2      ! Unless we are exiting a loop or an indirect command procedure, handle
2      ! the various looping constructs that have CIS entries.
2      !
2      IF NOT .EXIT_FLAG
2      THEN
3          BEGIN
3
3
3          ! If the top link is a FOR CIS, then increment the FOR-loop counter.
3          !
3          IF .dbg$gl_cishead[cis$b_input_type] EQL cis_for
3          THEN
4              BEGIN
```

```
;  704         0832  4
;  705         0833  4                bounds_match = FALSE;
;  706         0834  4
;  707         0835  4                ! Look up the loop counter.
;  708         0836  4                !
;  709         0837  4                varname = .dbg$gl_cishead [cis$a_for_loop_var];
;  710         0838  4                loop_incr = .dbg$gl_cishead [cis$l_for_loop_incr];
;  711         0839  4                IF dbg$def_sym_find (.varname, kind,
;  712         0840  4                                      valptr, global_flag, .message_vect)
;  713         0841  4                THEN
;  714         0842  5                    BEGIN
;  715         0843  5                    IF .kind EQL define_value
;  716         0844  5                    THEN
;  717         0845  6                        BEGIN
;  718         0846  6
;  719         0847  6                        value = .loop_incr + .valptr [dbg$l_value_value0];
;  720         0848  7                        IF (.loop_incr GTR 0
;  721         0849  7                           AND .value GTR .dbg$gl_cishead[cis$l_for_upper_bound])
;  722         0850  7                        OR (.loop_incr LSS 0
;  723         0851  7                           AND .value LSS .dbg$gl_cishead[cis$l_for_upper_bound])
;  724         0852  6                        THEN
;  725         0853  6                            bounds_match = TRUE
;  726         0854  6                        ELSE
;  727         0855  7                            BEGIN
;  728         0856  7
;  729         0857  7                            ! Copy the value descriptor. Fill in the new incremented
;  730         0858  7                            ! value into the copy. Save away the copy as the new
;  731         0859  7                            ! definition.
;  732         0860  7                            !
;  733         0861  7                            IF NOT dbg$nget_symid (.valptr, symid_list, .message_vect)
;  734         0862  7                            THEN
;  735         0863  7                                RETURN sts$k_severe;
;  736         0864  7                            IF NOT dbg$ncopy_desc (.valptr, new_valptr, .message_vect)
;  737         0865  7                            THEN
;  738         0866  7                                RETURN sts$k_severe;
;  739         0867  7                            dbg$sta_lock_symid (.symid_list);
;  740         0868  7                            new_valptr[dbg$l_value_value0] = .value;
;  741         0869  7                            ! Also copy the name.
;  742         0870  7                            new_name = dbg$get_memory (1+.varname[0]/4);
;  743         0871  7                            ch$move (1+.varname[0],.varname,.new_name);
;  744         0872  7                            IF NOT dbg$def_sym_add (.new_name, define_value,
;  745         0873  7                                            .new_valptr, FALSE, dummy, .message_vect)
;  746         0874  7                            THEN
;  747         0875  7                                RETURN sts$k_severe;
;  748         0876  7                            dbg$gl_cishead [cis$w_length] =
;  749         0877  7                                    .dbg$gl_cishead [cis$w_init_length];
;  750         0878  7                            dbg$gl_cishead [cis$a_input_ptr] =
;  751         0879  7                                .dbg$gl_cishead [cis$a_init_addr];
;  752         0880  7                            RETURN sts$k_success;
;  753         0881  6                            END;
;  754         0882  5                        END;
;  755         0883  4                    END;
;  756         0884  4
;  757         0885  4                ! Copy the loop variable name into temporary memory.
;  758         0886  4                ! This is for error-message purposes.
;  759         0887  4                !
;  760         0888  4                size = .varname[0];
```

```
                varname = dbg$get_tempmem (1+.size/4);
                ch$move (1+.size, .dbg$gl_cishead[cis$a_for_loop_var],
                                  .varname);

                ! If we fall through to here, we are exiting the loop for
                ! some reason.
                ! Release the space for the loop counter name.
                !
                dbg$rel_memory (.dbg$gl_cishead [cis$a_for_loop_var]);

                ! If bounds_match is false, we are exiting the loop not because
                ! the lower bound has matched the upper bound, but rather because
                ! the loop variable had been redefined.
                !
                IF NOT .bounds_match
                THEN
                    SIGNAL (dbg$_loopvar, 1, .varname);
                END;

            ! If the top link is a repeat cis, then decrement the count.
            !
            IF .dbg$gl_cishead[cis$b_input_type] EQL cis_repeat
            THEN
                BEGIN
                dbg$gl_cishead [cis$l_repeat_count] =
                    .dbg$gl_cishead [cis$l_repeat_count] - 1;

                ! If the repeat count is greater than zero, reset the cis
                ! to the beginning of the action buffer.
                !
                IF .dbg$gl_cishead [cis$l_repeat_count] GTR 0
                THEN
                    BEGIN
                    dbg$gl_cishead [cis$w_length] =
                        .dbg$gl_cishead [cis$w_init_length];
                    dbg$gl_cishead [cis$a_input_ptr] =
                        .dbg$gl_cishead [cis$a_init_addr];
                    RETURN sts$k_success;
                    END;

                END;

            END;


        ! If the top link is a WHILE, or a REPEAT whose count has gone to zero,
        ! an IF CIS, a FOR CIS, or a SCREEN CIS, then release storage for the
        ! action buffer.  Here we subtract two from the address because storage
        ! was allocated as a counted string and included the count word.
        !
        IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_WHILE  OR
           .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_REPEAT OR
           .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_IF     OR
           .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_FOR    OR
           .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_SCREEN
        THEN
            DBG$REL_MEMORY(.DBG$GL_CISHEAD[CIS$A_INIT_ADDR] - 2);
```

```


! If top link is a RAB, release the storage for the FAB, RAB and the
! buffer that holds the indirect command filespec.
!
IF .DBG$GL_CISHEAD[CIS$B_INPUT_TYPE] EQL CIS_RAB
THEN
    BEGIN

    LOCAL
        FAB_PTR : REF $FAB_DECL, ! File access block pointer
        RAB_PTR : REF $RAB_DECL; ! Record access block pointer

    RAB_PTR = .DBG$GL_CISHEAD [ CIS$A_INPUT_PTR];
    FAB_PTR = .RAB_PTR [RAB$L_FAB];
    IF .DBG$GB_DEF_OUT [OUT_VERIFY]
    THEN
        ICF_MESSAGE(2);      ! Exiting the ICF


    ! Release the filespec buffer. Remember this is a counted
    ! string so the address and length have to be adjusted to
    ! include the count.
    !
    DBG$REL_MEMORY (.FAB_PTR[FAB$L_FNA]-1);


    ! CLOSE and DISCONNECT
    !
    $CLOSE (FAB=.fab_ptr);

    dbg$rel_memory (.rab_ptr);
    dbg$rel_memory (.fab_ptr);

    ! Release the space taken up by the local define list.
    !
    IF NOT dbg$def_pr_exit (.message_vect)
    THEN
        RETURN sts$k_severe;

    END;

IF NOT .exit_flag
THEN
    BEGIN


    ! For a WHILE CIS, find out whether the condition is still true.
    !
    IF .dbg$gl_cishead [cis$b_input_type] EQL cis_while
    THEN
        BEGIN
        while_flag = TRUE;
        cond = .dbg$gl_cishead [cis$v_while_flag];
        END
    ELSE
        while_flag = FALSE;
```

```
;  875          1003  3
;  876          1004  2          END;
;  877          1005  2
;  878          1006  2
;  879          1007  2      ! Remove the link from the command input stream
;  880          1008  2      !
;  881          1009  2      temp = .dbg$gl_cishead ;
;  882          1010  2      dbg$gl_cishead = .dbg$gl_cishead [cis$a_next_link];
;  883          1011  2
;  884          1012  2      ! Now release the storage for the link itself
;  885          1013  2      !
;  886          1014  2      dbg$rel_memory (.temp);
;  887          1015  2
;  888          1016  2      IF NOT .exit_flag
;  889          1017  2      THEN
;  890          1018  2          ! If the cis is a WHILE, then set up the top cis for another iteration.
;  891          1019  2          !
;  892          1020  2          IF .while_flag
;  893          1021  2          THEN
;  894          1022  2              IF .cond
;  895          1023  2              THEN
;  896          1024  3                  BEGIN
;  897          1025  3                  dbg$gl_cishead [cis$a_input_ptr] =
;  898          1026  3                      .dbg$gl_cishead [cis$a_while_clause];
;  899          1027  3                  dbg$gl_cishead [cis$w_length] =
;  900          1028  3                      .dbg$gl_cishead [cis$w_while_length];
;  901          1029  2                  END;
;  902          1030  2
;  903          1031  2      RETURN sts$k_success;
;  904          1032  2
;  905          1033  1      END;



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

                                        08  00023 P.AAD:   .BYTE   8                                            ;
             67  6E  69  72  65  74  6E  65  00024          .ASCII  \entering\                                   ;
                                        07  0002C P.AAE:   .BYTE   7                                            ;
                 67  6E  69  74  69  78  65  0002D          .ASCII  \exiting\                                    ;

                                                  ENTER_PHRASE=        P.AAD
                                                  EXIT_PHRASE=         P.AAE
                                                           .EXTRN  SYS$CLOSE

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

                                      0FFC 00000           .ENTRY  DBG$NCIS_REMOVE, Save R2,R3,R4,R5,R6,R7,R8,-; 0718
                                                                   R9,R10,R11
                 5B 00000000G  00  9E 00002           MOVAB   DBG$GL_CISHEAD, R11                          ;
                 5E            18  C2 00009           SUBL2   #24, SP                                      ;
                    00000000G  00  D7 0000C           DECL    DBG$GL_CIS_LEVELS                            ; 0778
                 50            6B  D0 00012           MOVL    DBG$GL_CISHEAD, R0                           ; 0783
                 02       02   A0  91 00015           CMPB    2(R0), #2                                    ;
                               0A  12 00019           BNEQ    1$                                           ;
                          0C   A0  DD 0001B           PUSHL   12(R0)                                       ; 0785
        00000000G  00          01  FB 0001E           CALLS   #1, DBG$REL_MEMORY                           ;
```

```
                               50          6B  D0 00025 1$:     MOVL    DBG$GL_CISHEAD, R0                            ; 0792
                               52     30   A0  D0 00028         MOVL    48(R0), BUFLIST                               ;
                                           1B  13 0002C 2$:     BEQL    3$                                            ; 0793
                                      04   A2  DD 0002E         PUSHL   4(BUFLIST)                                    ; 0795
                    00000000G  00          01  FB 00031         CALLS   #1, DBG$REL_MEMORY                            ;
                               5A          62  D0 00038         MOVL    (BUFLIST), TEMP                               ; 0796
                                           52  DD 0003B         PUSHL   BUFLIST                                       ; 0797
                    00000000G  00          01  FB 0003D         CALLS   #1, DBG$REL_MEMORY                            ;
                               52          5A  D0 00044         MOVL    TEMP, BUFLIST                                 ; 0798
                                           E3  11 00047         BRB     2$                                            ; 0793
                               50          6B  D0 00049 3$:     MOVL    DBG$GL_CISHEAD, R0                            ; 0800
                                      30   A0  D4 0004C         CLRL    48(R0)                                        ;
                               08     02   A0  91 0004F         CMPB    2(R0), #8                                     ; 0809
                                           22  12 00053         BNEQ    4$                                            ;
00000000G  00        12   A0   01          02  EF 00055         EXTZV   #2, #1, 18(R0), DBG$GL_SCREEN_NOGO            ; 0812
                    00000000G  00     24   A0  D0 0005F         MOVL    36(R0), DBG$GL_SCREEN_OUTPUT                  ; 0813
                    00000000G  00     28   A0  D0 00067         MOVL    40(R0), DBG$GL_SCREEN_SOURCE                  ; 0814
                    00000000G  00     2C   A0  D0 0006F         MOVL    44(R0), DBG$GL_SCREEN_ERROR                   ; 0815
                               03     04   AC  E9 00077 4$:     BLBC    EXIT_FLAG, 5$                                 ; 0822
                                         0121  31 0007B         BRW     16$                                           ;
                               07     02   A0  91 0007E 5$:     CMPB    2(R0), #7                                     ; 0829
                                           03  13 00082         BEQL    6$                                            ;
                                         00FE  31 00084         BRW     14$                                           ;
                                           59  D4 00087 6$:     CLRL    BOUNDS_MATCH                                  ; 0833
                               57     1C   A0  D0 00089         MOVL    28(R0), VARNAME                               ; 0837
                               53     20   A0  D0 0008D         MOVL    32(R0), LOOP_INCR                             ; 0838
                                      08   AC  DD 00091         PUSHL   MESSAGE_VECT                                  ; 0840
                                      04   AE  9F 00094         PUSHAB  GLOBAL_FLAG                                   ; 0839
                                      0C   AE  9F 00097         PUSHAB  VALPTR                                        ;
                                      14   AE  9F 0009A         PUSHAB  KIND                                          ;
                                           57  DD 0009D         PUSHL   VARNAME                                       ;
                    00000000G  00          05  FB 0009F         CALLS   #5, DBG$DEF_SYM_FIND                          ;
                               2C          50  E9 000A6         BLBC    R0, 9$                                        ;
                               05     08   AE  D1 000A9         CMPL    KIND, #5                                      ; 0843
                                           26  12 000AD         BNEQ    9$                                            ;
                               52     04   AE  D0 000AF         MOVL    VALPTR, R2                                    ; 0847
                     54        53     20   A2  C1 000B3         ADDL3   32(R2), LOOP_INCR, VALUE                      ;
                                           53  D5 000B8         TSTL    LOOP_INCR                                     ; 0848
                                           09  15 000BA         BLEQ    7$                                            ;
                               50          6B  D0 000BC         MOVL    DBG$GL_CISHEAD, R0                            ; 0849
                          18   A0          54  D1 000BF         CMPL    VALUE, 24(R0)                                 ;
                                           0D  14 000C3         BGTR    8$                                            ;
                                           53  D5 000C5 7$:     TSTL    LOOP_INCR                                     ; 0850
                                           0E  18 000C7         BGEQ    10$                                           ;
                               50          6B  D0 000C9         MOVL    DBG$GL_CISHEAD, R0                            ; 0851
                          18   A0          54  D1 000CC         CMPL    VALUE, 24(R0)                                 ;
                                           05  18 000D0         BGEQ    10$                                           ;
                               59          01  D0 000D2 8$:     MOVL    #1, BOUNDS_MATCH                              ; 0853
                                           72  11 000D5 9$:     BRB     13$                                           ;
                                      08   AC  DD 000D7 10$:    PUSHL   MESSAGE_VECT                                  ; 0861
                                      10   AE  9F 000DA         PUSHAB  SYMID_LIST                                    ;
                                           52  DD 000DD         PUSHL   R2                                            ;
                    00000000G  00          03  FB 000DF         CALLS   #3, DBG$NGET_SYMID                            ;
                               55          50  E9 000E6         BLBC    R0, 11$                                       ;
                                      08   AC  DD 000E9         PUSHL   MESSAGE_VECT                                  ; 0864
                                      14   AE  9F 000EC         PUSHAB  NEW_VALPTR                                    ;
                                           52  DD 000EF         PUSHL   R2                                            ;
```

```
          00000000G  00           03  FB 000F1          CALLS   #3, DBG$NCOPY_DESC                       ;
                     43           50  E9 000F8          BLBC    R0, 11$                                  ;
                          0C      AE  DD 000FB          PUSHL   SYMID_LIST                               ; 0867
          00000000G  00           01  FB 000FE          CALLS   #1, DBG$STA_LOCK_SYMID                   ;
                     56   10      AE  D0 00105          MOVL    NEW_VALPTR, R6                           ; 0868
                 20  A6           54  D0 00109          MOVL    VALUE, 32(R6)                            ;
                     50           67  9A 0010D          MOVZBL  (VARNAME), R0                            ; 0870
                     50           04  C6 00110          DIVL2   #4, R0                                   ;
                          01      A0  9F 00113          PUSHAB  1(R0)                                    ;
          00000000G  00           01  FB 00116          CALLS   #1, DBG$GET_MEMORY                       ;
                     58           50  D0 0011D          MOVL    R0, NEW_NAME                             ;
                     50           67  9A 00120          MOVZBL  (VARNAME), R0                            ; 0871
                                  50  D6 00123          INCL    R0                                       ;
      68             67           50  28 00125          MOVC3   R0, (VARNAME), (NEW_NAME)                ;
                          08      AC  DD 00129          PUSHL   MESSAGE_VECT                             ; 0873
                          18      AE  9F 0012C          PUSHAB  DUMMY                                    ; 0872
                                  7E  D4 0012F          CLRL    -(SP)                                    ;
                                  56  DD 00131          PUSHL   R6                                       ; 0873
                                  05  DD 00133          PUSHL   #5                                       ; 0872
                                  58  DD 00135          PUSHL   NEW_NAME                                 ;
          00000000G  00           06  FB 00137          CALLS   #6, DBG$DEF_SYM_ADD                      ;
                     03           50  E8 0013E  11$:    BLBS    R0, 12$                                  ;
                               00F2  31 00141          BRW     20$                                      ;
                     50           6B  D0 00144  12$:    MOVL    DBG$GL_CISHEAD, R0                       ; 0876
                                  4A  11 00147          BRB     15$                                      ; 0877
                     52           67  9A 00149  13$:    MOVZBL  (VARNAME), SIZE                          ; 0888
      50             52           04  C7 0014C          DIVL3   #4, SIZE, R0                             ; 0889
                          01      A0  9F 00150          PUSHAB  1(R0)                                    ;
          00000000G  00           01  FB 00153          CALLS   #1, DBG$GET_TEMPMEM                      ;
                     57           50  D0 0015A          MOVL    R0, VARNAME                              ;
                                  52  D6 0015D          INCL    R2                                       ; 0890
                     56           6B  D0 0015F          MOVL    DBG$GL_CISHEAD, R6                       ;
      67         1C  B6           52  28 00162          MOVC3   R2, @28(R6), (VARNAME)                   ; 0891
                          1C      A6  DD 00167          PUSHL   28(R6)                                   ; 0897
          00000000G  00           01  FB 0016A          CALLS   #1, DBG$REL_MEMORY                       ;
                     11           59  E8 00171          BLBS    BOUNDS_MATCH, 14$                        ; 0903
                                  57  DD 00174          PUSHL   VARNAME                                  ; 0905
                                  01  DD 00176          PUSHL   #1                                       ;
                    000286C3      8F  DD 00178          PUSHL   #165571                                  ;
          00000000G  00           03  FB 0017E          CALLS   #3, LIB$SIGNAL                           ;
                     50           6B  D0 00185  14$:    MOVL    DBG$GL_CISHEAD, R0                       ; 0910
                     04   02      A0  91 00188          CMPB    2(R0), #4                                ;
                                  11  12 0018C          BNEQ    16$                                      ;
                          18      A0  D7 0018E          DECL    24(R0)                                   ; 0914
                                  0C  15 00191          BLEQ    16$                                      ; 0919
                     60   10      A0  B0 00193  15$:    MOVW    16(R0), (R0)                             ; 0923
                 04  A0   0C      A0  D0 00197          MOVL    12(R0), 4(R0)                            ; 0925
                               00DE  31 0019C          BRW     24$                                      ; 0926
                     50           6B  D0 0019F  16$:    MOVL    DBG$GL_CISHEAD, R0                       ; 0939
                     51   02      A0  9A 001A2          MOVZBL  2(R0), R1                                ;
                     05           51  91 001A6          CMPB    R1, #5                                   ;
                                  14  13 001A9          BEQL    17$                                      ;
                     04           51  91 001AB          CMPB    R1, #4                                   ; 0940
                                  0F  13 001AE          BEQL    17$                                      ;
                     06           51  91 001B0          CMPB    R1, #6                                   ; 0941
                                  0A  13 001B3          BEQL    17$                                      ;
                     07           51  91 001B5          CMPB    R1, #7                                   ; 0942
```

```
                                          05 13 001B8         BEQL    17$                           ;
                           08             51 91 001BA         CMPB    R1, #8                        ; 0943
                                          0C 12 001BD         BNEQ    18$                           ;
              7E       0C  A0             02 C3 001BF 17$:    SUBL3   #2, 12(R0), -(SP)             ; 0945
                00000000G  00             01 FB 001C4         CALLS   #1, DBG$REL_MEMORY            ;
                           50             6B D0 001CB 18$:    MOVL    DBG$GL_CISHEAD, R0            ; 0951
                           01         02  A0 91 001CE         CMPB    2(R0), #1                     ;
                                          66 12 001D2         BNEQ    21$                           ;
                           53         04  A0 D0 001D4         MOVL    4(R0), RAB_PTR                ; 0959
                           52         3C  A3 D0 001D8         MOVL    60(RAB_PTR), FAB_PTR          ; 0960
                           1F  00000000G  00 E9 001DC         BLBC    DBG$GB_DEF_OUT+2, 19$         ; 0961
                           50  00000000'  EF 9E 001E3         MOVAB   EXIT_PHRASE, PHRASE           ; 0963
                                      2C  A2 DD 001EA         PUSHL   44(FAB_PTR)                   ;
                           7E         34  A2 9A 001ED         MOVZBL  52(FAB_PTR), -(SP)            ;
                                          50 DD 001F1         PUSHL   PHRASE                        ;
                                          03 DD 001F3         PUSHL   #3                            ;
                                0002808B  8F DD 001F5         PUSHL   #163979                       ;
                00000000G  00             05 FB 001FB         CALLS   #5, DBG$NOUT_INFO             ;
              7E       2C  A2             01 C3 00202 19$:    SUBL3   #1, 44(FAB_PTR), -(SP)        ; 0970
                00000000G  00             01 FB 00207         CALLS   #1, DBG$REL_MEMORY            ;
                                          52 DD 0020E         PUSHL   FAB_PTR                       ; 0975
                00000000G  00             01 FB 00210         CALLS   #1, SYS$CLOSE                 ;
                                          53 DD 00217         PUSHL   RAB_PTR                       ; 0977
                00000000G  00             01 FB 00219         CALLS   #1, DBG$REL_MEMORY            ;
                                          52 DD 00220         PUSHL   FAB_PTR                       ; 0978
                00000000G  00             01 FB 00222         CALLS   #1, DBG$REL_MEMORY            ;
                                      08  AC DD 00229         PUSHL   MESSAGE_VECT                  ; 0982
                00000000G  00             01 FB 0022C         CALLS   #1, DBG$DEF_PR_EXIT           ;
                           04             50 E8 00233         BLBS    R0, 21$                       ;
                           50             04 D0 00236 20$:    MOVL    #4, R0                        ; 0984
                                             04 00239         RET                                   ;
                           16         04  AC E8 0023A 21$:    BLBS    EXIT_FLAG, 23$                ; 0988
                           50             6B D0 0023E         MOVL    DBG$GL_CISHEAD, R0            ; 0995
                           05         02  A0 91 00241         CMPB    2(R0), #5                     ;
                                          0B 12 00245         BNEQ    22$                           ;
                           52             01 D0 00247         MOVL    #1, WHILE_FLAG                , 0998
53       12   A0           01             01 EF 0024A         EXTZV   #1, #1, 18(R0), COND          ; 0999
                                          02 11 00250         BRB     23$                           ; 0995
                                          52 D4 00252 22$:    CLRL    WHILE_FLAG                    ; 1002
                           50             6B D0 00254 23$:    MOVL    DBG$GL_CISHEAD, R0            ; 1009
                           5A             50 D0 00257         MOVL    R0, TEMP                      ;
                           6B         08  A0 D0 0025A         MOVL    8(R0), DBG$GL_CISHEAD         ; 1010
                                          5A DD 0025E         PUSHL   TEMP                          ; 1014
                00000000G  00             01 FB 00260         CALLS   #1, DBG$REL_MEMORY            ;
                           12         04  AC E8 00267         BLBS    EXIT_FLAG, 24$                ; 1016
                           0F             52 E9 0026B         BLBC    WHILE_FLAG, 24$               ; 1020
                           0C             53 E9 0026E         BLBC    COND, 24$                     ; 1022
                           50             6B D0 00271         MOVL    DBG$GL_CISHEAD, R0            ; 1025
                       04  A0         14  A0 D0 00274         MOVL    20(R0), 4(R0)                 ; 1026
                           60         34  A0 B0 00279         MOVW    52(R0), (R0)                  ; 1028
                           50             01 D0 0027D 24$:    MOVL    #1, R0                        ; 1031
                                             04 00280         RET                                   ; 1033

; Routine Size:  641 bytes,    Routine Base: DBG$CODE + 031E
```

```
1 GLOBAL ROUTINE DBG$NGET_ADDRESS (ADDR_EXP_DESC, ADDRESS, TYPE, PROLOG_FLAG, MESSAGE_VECT) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is called with a descriptor, as returned
1 !       by the Address Expression Interpreter, to obtain the address bound to the
1 !       entity described by the descriptor.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ADDR_EXP_DESC   - A longword containing the address of either a
1 !                         value or primary descriptor
1 !
1 !       ADDRESS         - The address of a quadword to contain the resulting
1 !                         byte address and bit offset
1 !
1 !       TYPE            - The address of a longword to contain the type of the address
1 !                         (No longer used).
1 !
1 !       PROLOG_FLAG     - A flag set to true to indicate this routine is
1 !                         called from SET BREAK/TRACE, SHOW BREAK/TRACE, where
1 !                         routine break address is taken from the primary
1 !                         routine/entry rst entry.
1 !
1 !       MESSAGE_VECT    - The address of a longword to contain the address of a
1 !                         message argument vector upon detection of errors
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       On error, a message argument vector is constructed and returned.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       An unsigned integer longword completion code
1 !
1 ! COMPLETION CODES:
1 !
1 !       STS$K_SUCCESS (1)       - Success. Address and type returned.
1 !
1 !       STS$K_SEVERE  (4)       - Failure. No type and/or address obtained.
1 !                                 Message argument vector returned.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2     MAP
2       ADDRESS: REF VECTOR[,LONG],
2       ADDR_EXP_DESC: REF DBG$VALDESC; ! Points to a new style Descriptor.
2
```

```
LOCAL
    VMS_DESC: REF DBG$STG_DESC,
    RSTPTR: REF RST$ENTRY;


! If the flag is set, take the break address from Routine/Entry RST
! in Primary.  (The only way this flag can be set is in DBGEVENT.)
!
IF .PROLOG_FLAG
THEN
    BEGIN
    RSTPTR = .ADDR_EXP_DESC[DBG$L_DHDR_SYMID0];
    ADDRESS[0] = .RSTPTR[RST$L_BREAKADDR];
    ADDRESS[1] = 0,
    RETURN sts$k_success;
    END;


! Check whether we are looking at a Primary Descriptor.
!
IF  .ADDR_EXP_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC
THEN
    BEGIN


    ! Allocate temporary memory for the VMS descriptor.
    !
    VMS_DESC = DBG$GET_TEMPMEM (3);

    ! Call the routine that fills in the VMS descriptor.
    !
    DBG$MAKE_VMS_DESC (.ADDR_EXP_DESC, .VMS_DESC);
    END

! Check for Volatile Value Descriptor.
!
ELSE
    IF .ADDR_EXP_DESC [DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
    THEN
        VMS_DESC = ADDR_EXP_DESC [DBG$A_VALUE_VMSDESC]

    ! Any other kind of descriptor is an error.
    !
    ELSE
        $DBG_ERROR ('DBGNEXCTE\DBG$NGET_ADDRESS unexpected descriptor type');

! Fill in the output parameter to point to the
! (byte address, bit offset) quadword in the VMS descriptor.
!
ADDRESS[0] = .VMS_DESC[DSC$A_POINTER];
IF .VMS_DESC[DSC$B_CLASS] NEQ DSC$K_CLASS_UBS
THEN
    ADDRESS[1] = 0
ELSE
    ADDRESS[1] = .VMS_DESC[DSC$L_POS];

RETURN sts$k_success;
```

```
; 1021          1148  1     END;                  . End of dbg$nget_address


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

24  47  42  44  5C  45  54  43  58  45  4E  47  42  44  35  00034 P.AAF:  .ASCII  \5DBGNEXCTE\<92>\DBG$NGET_ADDRESS unexpe\   ;
6E  75  20  53  53  45  52  44  44  41  5F  54  45  47  4E  00043                                                             ;
                                            65  70  78  65  00052                                                             ;
72  6F  74  70  69  72  63  73  65  64  20  64  65  74  63  00056          .ASCII  \cted descriptor type\                    ;
                                        65  70  79  74  20  00065                                                             ;


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

                                                          0004 00000         .ENTRY  DBG$NGET_ADDRESS, Save R2            ; 1034
                                   12        10      AC   E9 00002           BLBC    PROLOG_FLAG, 1$                      ; 1099
                                   50        04      AC   D0 00006           MOVL    ADDR_EXP_DESC, R0                    ; 1102
                                   51        0C      A0   D0 0000A           MOVL    12(R0), RSTPTR                       ;
                                   50        08      AC   D0 0000E           MOVL    ADDRESS, R0                          ; 1103
                                   60        28      A1   D0 00012           MOVL    40(RSTPTR), (R0)                     ;
                                                     5D   11 00016           BRB     5$                                   ; 1104
00000079   8F       04   BC        08                10   ED 00018 1$:       CMPZV   #16, #8, @ADDR_EXP_DESC, #121        ; 1111
                                                     1A   12 00022           BNEQ    2$                                   ;
                                                     03   DD 00024           PUSHL   #3                                   ; 1118
                        00000000G  00                01   FB 00026           CALLS   #1, DBG$GET_TEMPMEM                  ;
                                   52                50   D0 0002D           MOVL    R0, VMS_DESC                         ;
                                                     52   DD 00030           PUSHL   VMS_DESC                             ; 1122
                                             04      AC   DD 00032           PUSHL   ADDR_EXP_DESC                        ;
                        00000000G  00                02   FB 00035           CALLS   #2, DBG$MAKE_VMS_DESC                ;
                                                     28   11 0003C           BRB     4$                                   ; 1111
00000083   8F       04   BC        08                10   ED 0003E 2$:       CMPZV   #16, #8, @ADDR_EXP_DESC, #131        ; 1128
                                                     07   12 00048           BNEQ    3$                                   ;
                    52       04   AC                 14   C1 0004A           ADDL3   #20, ADDR_EXP_DESC, VMS_DESC         ; 1130
                                                     15   11 0004F           BRB     4$                                   ;
                                  00000000'          EF   9F 00051 3$:       PUSHAB  P.AAF                                ; 1135
                                                     01   DD 00057           PUSHL   #1                                   ;
                                  00028362           8F   DD 00059           PUSHL   #164706                              ;
                        00000000G  00                03   FB 0005F           CALLS   #3, LIB$SIGNAL                       ;
                              08   BC        04      A2   D0 00066 4$:       MOVL    4(VMS_DESC), @ADDRESS                ; 1140
                                   50        08      AC   D0 0006B           MOVL    ADDRESS, R0                          ; 1143
                                   0D        03      A2   91 0006F           CMPB    3(VMS_DESC), #13                     ; 1141
                                                     05   13 00073           BEQL    6$                                   ;
                                             04      A0   D4 00075 5$:       CLRL    4(R0)                                ; 1143
                                                     05   11 00078           BRB     7$                                   ;
                              04   A0        08      A2   D0 0007A 6$:       MOVL    8(VMS_DESC), 4(R0)                   ; 1145
                                   50                01   D0 0007F 7$:       MOVL    #1, R0                               ; 1147
                                                          04 00082           RET                                          ; 1148

; Routine Size:  131 bytes,    Routine Base:  DBG$CODE + 059F


; 1022          1149  1 END                                 !End of module
; 1023          1150  0 ELUDOM
```

```
                                                                    .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;       Name                     Bytes                          Attributes
;
;  DBG$PLIT                        106  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                       1570  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        23         0        1000      00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1        32         0         0           7      00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1        1545       167        10          97      00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                              418         0         0          31      00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1         386         4         1          22      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1         150         2         1          12      00:00.3



;                                    COMMAND QUALIFIERS

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

; Size:          1570 code + 106 data bytes
; Run Time:         00:32.2
; Elapsed Time:     01:41.7
; Lines/CPU Min:      2146
; Lexemes/CPU-Min: 12100
; Memory Used:  261 pages
; Compilation Complete
```

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGNERMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS