F 1

```
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBBB    UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBBBB    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**DBGNPARSE

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
DDDDDDDD    BBBBBBBB      GGGGGGGG  NN      NN  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
DD      DD  BB      BB  GG          NN      NN  PP      PP  AA      AA  RR      RR  SS          EE
DD      DD  BB      BB  GG          NN      NN  PP      PP  AA      AA  RR      RR  SS          EE
DD      DD  BB      BB  GG          NNNN    NN  PP      PP  AA      AA  RR      RR  SS          EE
DD      DD  BB      BB  GG          NNNN    NN  PP      PP  AA      AA  RR      RR  SS          EE
DD      DD  BBBBBBBB    GG          NN  NN  NN  PPPPPPPP    AA      AA  RRRRRRRR      SSSSSS    EEEEEEEE
DD      DD  BBBBBBBB    GG          NN  NN  NN  PPPPPPPP    AA      AA  RRRRRRRR      SSSSSS    EEEEEEEE
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  PP          AAAAAAAAAA  RR  RR              SS  EE
DD      DD  BB      BB  GG  GGGGGG  NN    NNNN  PP          AAAAAAAAAA  RR  RR              SS  EE
DD      DD  BB      BB  GG      GG  NN      NN  PP          AA      AA  RR    RR            SS  EE              ....
DD      DD  BB      BB  GG      GG  NN      NN  PP          AA      AA  RR    RR            SS  EE              ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  PP          AA      AA  RR      RR  SSSSSSSS    EEEEEEEEEE      ....
DDDDDDDD    BBBBBBBB      GGGGGG    NN      NN  PP          AA      AA  RR      RR  SSSSSSSS    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
```

```
MODULE DBGNPARSE (IDENT = 'V04-000') =

BEGIN
!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!
! WRITTEN BY
!       David Plummer   April 15, 1980
!
! MODIFIED BY
!       Rich Title
!       Vicki Holt
!       Bert Beander
!
! MODULE FUNCTION
!       This module contains the highest level parse network, DBG$NPARSE_CMD, and
!       several parsing associated routines. Legal command verbs are recognized by
!       DBG$NPARSE_CMD and control is passed to the subnetwork responsible for parsing
!       the rest of the input associated with the verb recognized. In total, the
!       routines produce a command execution tree which is the version 3 debugger
!       form of intermediate code.
!
!       Also contained in this module are the routines DBG$NPARSE_ADDRESS and
!       DBG$NPARSE_EXPRESSION which are the interfaces between the debugger
!       parser and the Address Expression Interpreter and Expression Interpreter.
!

REQUIRE 'SRC$:DBGPROLOG.REQ';

FORWARD ROUTINE
    DBG$NPARSE_CMD,                     ! Highest level network for parsing
    DBG$NMATCH,                         ! String matching routine
    DBG$NNEXT_WORD,                     ! Produces a counted string from the input
    DBG$NPARSE_EXPRESSION,              ! Interface to expression interpreters
    DBG$NPARSE_ADDRESS,                 ! Interface to address expression interpreters
```

```
;     58          0191   1      DBG$NSAVE_DECIMAL_INTEGER,          ! Converts ASCII input to integer
;     59          0192   1      DBG$NSAVE_STRING,                   ! Stores a string from input
;     60          0193   1      DBG$NGET_DIR_LIST,                  ! Parses a directory list.
;     61          0194   1      DBG$EXPAND_DEFINE_NAME,             ! Expands name defined with DEFINE
;     62          0195   1      DBG$SYNTAX_ERROR: NOVALUE;          ! Signal a syntax error in command
```

```
EXTERNAL ROUTINE
	DBG$ADDR_EXP_INT,			! Address Expression Interpreter
	DBG$DEF_SYM_FIND,			! Look up a symbol in the
						!     DEFINE symbol table
	DBG$GET_MEMORY,				! Allocates space
	DBG$GET_TEMPMEM,			! Allocates and lists dynamic storage
	DBG$EXP_INT,				! Expression Interpreter
	DBG$NPARSE_ALLOCATE,			! ALLOCATE command parse network
	DBG$NPARSE_AT_SIGN,			! @ filespec parse network
	DBG$NPARSE_ATTACH,			! ATTACH command parse network
	DBG$NPARSE_CALL,			! CALL command parse network
	DBG$NPARSE_CANCEL,			! CANCEL command parse network
	DBG$NPARSE_DECLARE,			! DECLARE command parse network
	DBG$NPARSE_DEFINE,			! DEFINE command parse network
	DBG$NPARSE_DELETE,			! DELETE/KEY command
	DBG$NPARSE_DEPOSIT,			! DEPOSIT command parse network
	DBG$NPARSE_DUMP,			! DUMP command parse network
	DBG$NPARSE_EDIT,			! EDIT command parse network
	DBG$NPARSE_EVALUATE,			! EVALUATE command parse network
	DBG$NPARSE_EXAMINE,			! EXAMINE command parse network
	DBG$NPARSE_EXIT,			! EXIT command parse network
	DBG$NPARSE_EXITLOOP,			! EXITLOOP command parse network
	DBG$NPARSE_FOR,				! FOR command parse network
	DBG$NPARSE_GO,				! GO command parse network
	DBG$NPARSE_HELP,			! HELP command parse network
	DBG$NPARSE_IF,				! IF command parse network
	DBG$NPARSE_REPEAT,			! REPEAT command parse network
	DBG$NPARSE_SEARCH,			! SEARCH command parse network
	DBG$NPARSE_SET,				! SET command parse network
	DBG$NPARSE_SHOW,			! SHOW command parse network
	DBG$NPARSE_SPAWN,			! SPAWN command parse network
	DBG$NPARSE_STEP,			! STEP command parse network
	DBG$NPARSE_SYMBOLIZE,			! SYMBOLIZE command parse network
	DBG$NPARSE_TYPE,			! TYPE command parse network
	DBG$NPARSE_UNDEFINE,			! UNDEFINE command parse network
	DBG$NPARSE_WHILE,			! WHILE command parse network
	DBG$NMAKE_ARG_VECT,			! Constructs a message vector
	DBG$NOUT_INFO,				! Outputs an info message
	DBG$NREAD_NAME,				! Reads a name that is
						!     a potential DEFINEd name
	DBG$NSET_LAST_TYPLEN,			! Sets type and length of current location
	DBG$NSYNTAX_ERROR,			! Constructs a message vector for a syntax error
	DBG$REL_MEMORY,				! Release memory from pool
	DBG$SCR_PARSE_DISPLAY_CMD: NOVALUE,	! Parse the DISPLAY command
	DBG$SCR_PARSE_SAVE_CMD: NOVALUE,	! Parse the SAVE command
	DBG$SCR_PARSE_SCROLL_CMD: NOVALUE,	! Parse the SCROLL command
	DBG$SCR_PARSE_SELECT_CMD: NOVALUE,	! parse the SELECT command
	DBG$STA_SETCONTEXT : NOVALUE;		! Sets registers context

EXTERNAL
	DBG$GL_DEVELOPER: BITVECTOR[],		! Developer flags
	DBG$GL_EDIT_ENABLED,			! Flag saying whether the EDIT
						!     command is enabled.
	DBG$GB_RADIX: VECTOR[3, BYTE],		! Radix settings
	DBG$GB_EXC_BRE_FLAG: BYTE,		! Flag set if we are in exception break
	DBG$GB_LANGUAGE: BYTE,			! Current language code
	DBG$GL_SCREEN_NOGO,			! Flag to disable STEP, GO, and CALL in
```

```
1                                               !       screen display command lists
1       DBG$GB_VERB: BYTE;                      ! Holds command verb value
1
1 LITERAL
1       ALLOCATE_VERB = DBG$K_ALLOCATE_VERB,    ! Code for ALLOCATE command
1       AT_SIGN_VERB = DBG$K_AT_SIGN_VERB,      ! Code for indirect command file execution
1       ATTACH_VERB = DBG$K_ATTACH_VERB,        ! Code for ATTACH command
1       CALL_VERB = DBG$K_CALL_VERB,            ! Code for CALL command
1       CANCEL_VERB = DBG$K_CANCEL_VERB,        ! Code for CANCEL command
1       DECLARE_VERB = DBG$K_DECLARE_VERB,      ! Code for DECLARE command
1       DEFINE_VERB = DBG$K_DEFINE_VERB,        ! Code for DEFINE command
1       DELETE_VERB = DBG$K_DELETE_VERB,        ! Code for DELETE command
1       DEPOSIT_VERB = DBG$K_DEPOSIT_VERB,      ! Code for DEPOSIT command
1       DISPLAY_VERB = DBG$K_DISPLAY_VERB,      ! Code for DISPLAY command
1       DUMP_VERB = DBG$K_DUMP_VERB,            ! Code for DUMP command
1       EDIT_VERB = DBG$K_EDIT_VERB,            ! Code for EDIT command
1       EVALUATE_VERB = DBG$K_EVALUATE_VERB,    ! Code for EVALUATE command
1       EXAMINE_VERB = DBG$K_EXAMINE_VERB,      ! Code for EXAMINE command
1       EXIT_VERB = DBG$K_EXIT_VERB,            ! Code for EXIT command
1       EXITLOOP_VERB = DBG$K_EXITLOOP_VERB,    ! Code for EXITLOOP command
1       FOR_VERB = DBG$K_FOR_VERB,              ! Code for FOR command
1       GO_VERB = DBG$K_GO_VERB,                ! Code for GO command
1       HELP_VERB = DBG$K_HELP_VERB,            ! Code for HELP command
1       IF_VERB = DBG$K_IF_VERB,                ! Code for IF command
1       REPEAT_VERB = DBG$K_REPEAT_VERB,        ! Code for REPEAT command
1       SAVE_VERB = DBG$K_SAVE_VERB,            ! Code for SAVE command
1       SCROLL_VERB = DBG$K_SCROLL_VERB,        ! Code for SCROLL command
1       SEARCH_VERB = DBG$K_SEARCH_VERB,        ! Code for SEARCH command
1       SELECT_VERB = DBG$K_SELECT_VERB,        ! Code for SELECT command
1       SET_VERB = DBG$K_SET_VERB,              ! Code for SET command
1       SHOW_VERB = DBG$K_SHOW_VERB,            ! Code for SHOW command
1       SPAWN_VERB = DBG$K_SPAWN_VERB,          ! Code for SPAWN command
1       STEP_VERB = DBG$K_STEP_VERB,            ! Code for STEP command
1       SYMBOLIZE_VERB =
1                   DBG$K_SYMBOLIZE_VERB,       ! Code for SYMBOLIZE command
1       TYPE_VERB = DBG$K_TYPE_VERB,            ! Code for TYPE command
1       UNDEFINE_VERB = DBG$K_UNDEFINE_VERB,    ! Code for UNDEFINE command
1       WHILE_VERB = DBG$K_WHILE_VERB,          ! Code for WHILE command
1       WORD_SIZE = 80;                         ! Maximum word size
1
1 OWN
1       WORD_BUF : VECTOR [WORD_SIZE + 1, BYTE]; ! Buffer for counted string word
```

```
GLOBAL ROUTINE DBG$NPARSE_CMD (INPUT_DESC, VERB_NODE_PTR, MESSAGE_VECT) =
!
! FUNCTIONAL DESCRIPTION:
!
!       Highest level command parsing ATN network. This routine recognizes the
!       verb portion of the input command and transfers control to the appropriate
!       ATN subnetwork to parse the rest of the command.
!
! FORMAL PARAMETERS:
!
!       INPUT_DESC      - Standard VAX string descriptor of the input command.
!
!       VERB_NODE_PTR   - Pointer to the verb (head) node of the command
!                         execution tree.
!
!       MESSAGE_VECT    - Address of a longword to contain the address of
!                         a message argument vector.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       The command execution tree is constructed and verb_node_ptr is set to point
!       to the dynamically allocated verb node which is the head node of the tree.
!
! ROUTINE VALUE:
!
!       unsigned longword integer completion code
!
! COMPLETION CODES:
!
!       STS$K_SEVERE (4) -      unsuccessful parse
!
!       STS$K_SUCCESS (1) -     successful parse of the input command
!

    BEGIN

    MAP
        INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor

    ! Define strings used at this level of parsing
    !
    BIND
        DBG$CS_ALLOCATE  = UPLIT BYTE(%ASCIC 'ALLOCATE'),
        DBG$CS_AT_SIGN   = UPLIT BYTE(1, DBG$K_AT_SIGN),
        DBG$CS_ATTACH    = UPLIT BYTE(%ASCIC 'ATTACH'),
        DBG$CS_CALL      = UPLIT BYTE(%ASCIC 'CALL'),
        DBG$CS_CANCEL    = UPLIT BYTE(%ASCIC 'CANCEL'),
        DBG$CS_DECLARE   = UPLIT BYTE(%ASCIC 'DECLARE'),
        DBG$CS_DEFINE    = UPLIT BYTE(%ASCIC 'DEFINE'),
        DBG$CS_DELETE    = UPLIT BYTE(%ASCIC 'DELETE'),
        DBG$CS_DEPOSIT   = UPLIT BYTE(%ASCIC 'DEPOSIT'),
        DBG$CS_DISPLAY   = UPLIT BYTE(%ASCIC 'DISPLAY'),
```

```
          DBG$CS_DUMP       = UPLIT BYTE(%ASCIC 'DUMP'),
          DBG$CS_EDIT       = UPLIT BYTE(%ASCIC 'EDIT'),
          DBG$CS_EVALUATE   = UPLIT BYTE(%ASCIC 'EVALUATE'),
          DBG$CS_EXAMINE    = UPLIT BYTE(%ASCIC 'EXAMINE'),
          DBG$CS_EXIT       = UPLIT BYTE(%ASCIC 'EXIT'),
          DBG$CS_EXITLOOP   = UPLIT BYTE(%ASCIC 'EXITLOOP'),
          DBG$CS_FOR        = UPLIT BYTE(%ASCIC 'FOR'),
          DBG$CS_GO         = UPLIT BYTE(%ASCIC 'GO'),
          DBG$CS_HELP       = UPLIT BYTE(%ASCIC 'HELP'),
          DBG$CS_IF         = UPLIT BYTE(%ASCIC 'IF'),
          DBG$CS_REPEAT     = UPLIT BYTE(%ASCIC 'REPEAT'),
          DBG$CS_SAVE       = UPLIT BYTE(%ASCIC 'SAVE'),
          DBG$CS_SCROLL     = UPLIT BYTE(%ASCIC 'SCROLL'),
          DBG$CS_SEARCH     = UPLIT BYTE(%ASCIC 'SEARCH'),
          DBG$CS_SELECT     = UPLIT BYTE(%ASCIC 'SELECT'),
          DBG$CS_SET        = UPLIT BYTE(%ASCIC 'SET'),
          DBG$CS_SHOW       = UPLIT BYTE(%ASCIC 'SHOW'),
          DBG$CS_SPAWN      = UPLIT BYTE(%ASCIC 'SPAWN'),
          DBG$CS_STEP       = UPLIT BYTE(%ASCIC 'STEP'),
          DBG$CS_SYMBOLIZE  = UPLIT BYTE(%ASCIC 'SYMBOLIZE'),
          DBG$CS_TYPE       = UPLIT BYTE(%ASCIC 'TYPE'),
          DBG$CS_UNDEFINE   = UPLIT BYTE(%ASCIC 'UNDEFINE'),
          DBG$CS_WHILE      = UPLIT BYTE(%ASCIC 'WHILE'),
          DBG$CS_CR         = UPLIT BYTE(1, DBG$K_CAR_RETURN);

    LOCAL
          VERB_NODE : REF DBG$VERB_NODE;  ! Verb node (head of tree)

    !
    ! Construct and link the verb node
    !
    VERB_NODE = DBG$GET_TEMPMEM(DBG$K_VERB_NODE_SIZE);
    .VERB_NODE_PTR = .VERB_NODE;
    DBG$GB_VERB = 0;

    !
    ! Set registers to current context as a default
    !
    DBG$STA_SETCONTEXT (0);

    !
    ! Try to recognize a legal verb. If one is found, transfer control
    ! to a command parse subnetwork.
    !
    IF (SELECTONE TRUE OF
            SET

            [DBG$NMATCH (.INPUT_DESC, DBG$CS_ALLOCATE, 2)]:
                BEGIN
                VERB_NODE [DBG$B_VERB_LITERAL] = ALLOCATE_VERB;
                DBG$GB_VERB = DBG$K_ALLOCATE_VERB;
                DBG$NPARSE_ALLOCATE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
                END;

            [DBG$NMATCH (.INPUT_DESC, DBG$CS_AT_SIGN, 1)]:
                BEGIN
                VERB_NODE [DBG$B_VERB_LITERAL] = AT_SIGN_VERB;
```

```
;   4              DBG$GB_VERB = DBG$K_AT_SIGN_VERB;
;   4              DBG$NPARSE_AT_SIGN (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
;   3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_ATTACH, 3)]:
;   4              BEGIN
;   4              VERB_NODE [DBG$B_VERB_LITERAL] = ATTACH_VERB;
;   4              DBG$GB_VERB = DBG$K_ATTACH_VERB;
;   4              DBG$NPARSE_ATTACH (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
;   3
;   3          ! Check for the CALL command.  If this is the CALL command, we
;   3          ! signal an error if we are currently in an exception break
;   3          ! (we cannot do a CALL if we will restart the program by re-
;   3          ! signalling instead of continuing) or if we are currently
;   3          ! executing the DEBUG command list of a screen display (which
;   3          ! would cause an infinite loop of screen display updating).
;   3          ! Otherwise we parse the command normally.
;   3          !
;   3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_CALL, 3)]:
;   4              BEGIN
;   4              VERB_NODE [DBG$B_VERB_LITERAL] = CALL_VERB;
;   4              DBG$GB_VERB = DBG$K_CALL_VERB;
;   4              IF .DBG$GB_EXC_BRE_FLAG THEN SIGNAL(DBG$_STEFROEXC);
;   4              IF .DBG$GL_SCREEN_NOGO THEN SIGNAL(DBG$_NOSTEPGO);
;   4              DBG$NPARSE_CALL (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
;   3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_CANCEL, 3)]:
;   4              BEGIN
;   4              VERB_NODE [DBG$B_VERB_LITERAL] = CANCEL_VERB;
;   4              DBG$GB_VERB = DBG$K_CANCEL_VERB;
;   4              DBG$NPARSE_CANCEL (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
;   3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DECLARE, 3)]:
;   4              BEGIN
;   4              VERB_NODE [DBG$B_VERB_LITERAL] = DECLARE_VERB;
;   4              DBG$GB_VERB = DBG$K_DECLARE_VERB;
;   4              DBG$NPARSE_DECLARE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
;   3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEFINE, 3)]:
;   4              BEGIN
;   4              VERB_NODE [DBG$B_VERB_LITERAL] = DEFINE_VERB;
;   4              DBG$GB_VERB = DBG$K_DEFINE_VERB;
;   4              DBG$NPARSE_DEFINE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
;   3          [DBG$NMATCH (.INPUT_DESC, DBG$CS_DELETE, 3)]:
;   4              BEGIN
;   4              VERB_NODE [DBG$B_VERB_LITERAL] = DELETE_VERB;
;   4              DBG$GB_VERB = DBG$K_DELETE_VERB;
;   4              DBG$NPARSE_DELETE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
;   3              END;
;   3
```

```
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_DEPOSIT, 1)]:
4                BEGIN
4                VERB_NODE [DBG$B_VERB_LITERAL] = DEPOSIT_VERB;
4                DBG$GB_VERB = DBG$K_DEPOSIT_VERB;
4                DBG$NPARSE_DEPOSIT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3                END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_DISPLAY, 3)]:
4                BEGIN
4                VERB_NODE [DBG$B_VERB_LITERAL] = DISPLAY_VERB;
4                DBG$GB_VERB = DBG$K_DISPLAY_VERB;
4                DBG$SCR_PARSE_DISPLAY_CMD(.INPUT_DESC, FALSE, .VERB_NODE);
4                TRUE
3                END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_DUMP, 2)]:
4                BEGIN
4                VERB_NODE [DBG$B_VERB_LITERAL] = DUMP_VERB;
4                DBG$GB_VERB = DBG$K_DUMP_VERB;
4                DBG$NPARSE_DUMP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3                END;
3
3
3           ! The EDIT command is our interface to the EDITH text editor.
3           ! We allow this command if a certain global flag is set.
3           ! That way, the command can be disabled for version 4.0,
3           ! and enabled with a patch in a future release. The
3           ! all-purpose developer bit 0 also enables this command,
3           ! so that developers can easily test it.
3           !
4           [  (IF .DBG$GL_EDIT_ENABLED OR .DBG$GL_DEVELOPER[0]
4                THEN
4                    DBG$NMATCH (.INPUT_DESC, DBG$CS_EDIT, 2)
4                ELSE
3                    FALSE)]:
4                BEGIN
4                VERB_NODE [DBG$B_VERB_LITERAL] = EDIT_VERB;
4                DBG$GB_VERB = DBG$K_EDIT_VERB;
4                DBG$NPARSE_EDIT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3                END;
3
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_EVALUATE, 2)]:
4                BEGIN
4                VERB_NODE [DBG$B_VERB_LITERAL] = EVALUATE_VERB;
4                DBG$GB_VERB = DBG$K_EVALUATE_VERB;
4                DBG$NPARSE_EVALUATE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3                END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXAMINE, 1)]:
4                BEGIN
4                VERB_NODE [DBG$B_VERB_LITERAL] = EXAMINE_VERB;
4                DBG$GB_VERB = DBG$K_EXAMINE_VERB;
4                DBG$NPARSE_EXAMINE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3                END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXIT, 3)]:
```

```
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = EXIT_VERB;
            DBG$GB_VERB = DBG$K_EXIT_VERB;
            DBG$NPARSE_EXIT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_EXITLOOP, 5)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = EXITLOOP_VERB;
            DBG$GB_VERB = DBG$K_EXITLOOP_VERB;
            DBG$NPARSE_EXITLOOP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_FOR, 1)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = FOR_VERB;
            DBG$GB_VERB = DBG$K_FOR_VERB;
            DBG$NPARSE_FOR (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_GO, 1)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = GO_VERB;
            DBG$GB_VERB = DBG$K_GO_VERB;
            IF .DBG$GL_SCREEN_NOGO THEN SIGNAL(DBG$_NOSTEPGO);
            DBG$NPARSE_GO (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_HELP, 1)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = HELP_VERB;
            DBG$GB_VERB = DBG$K_HELP_VERB;
            DBG$NPARSE_HELP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_IF, 1)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = IF_VERB;
            DBG$GB_VERB = DBG$K_IF_VERB;
            DBG$NPARSE_IF (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_REPEAT, 2)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = REPEAT_VERB;
            DBG$GB_VERB = DBG$K_REPEAT_VERB;
            DBG$NPARSE_REPEAT (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
            END;

        [DBG$NMATCH (.INPUT_DESC, DBG$CS_SAVE, 3)]:
            BEGIN
            VERB_NODE [DBG$B_VERB_LITERAL] = SAVE_VERB;
            DBG$GB_VERB = DBG$K_SAVE_VERB;
            DBG$SCR_PARSE_SAVE_CMD(.INPUT_DESC, .VERB_NODE);
            TRUE
            END;
```

```
[DBG$NMATCH (.INPUT_DESC, DBG$CS_SCROLL, 3)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SCROLL_VERB;
    DBG$GB_VERB = DBG$K_SCROLL_VERB;
    DBG$SCR_PARSE_SCROLL_CMD(.INPUT_DESC, .VERB_NODE);
    TRUE
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_SEARCH, 3)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SEARCH_VERB;
    DBG$GB_VERB = DBG$K_SEARCH_VERB;
    DBG$NPARSE_SEARCH (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_SELECT, 3)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SELECT_VERB;
    DBG$GB_VERB = DBG$K_SELECT_VERB;
    DBG$SCR_PARSE_SELECT_CMD(.INPUT_DESC, .VERB_NODE);
    TRUE
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_SET, 2)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SET_VERB;
    DBG$GB_VERB = DBG$K_SET_VERB;
    DBG$NPARSE_SET (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_SHOW, 2)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SHOW_VERB;
    DBG$GB_VERB = DBG$K_SHOW_VERB;
    DBG$NPARSE_SHOW (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_SPAWN, 2)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SPAWN_VERB;
    DBG$GB_VERB = DBG$K_SPAWN_VERB;
    DBG$NPARSE_SPAWN (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_STEP, 1)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = STEP_VERB;
    DBG$GB_VERB = DBG$K_STEP_VERB;
    IF .DBG$GL_SCREEN_NOGO THEN SIGNAL(DBG$_NOSTEPGO);
    DBG$NPARSE_STEP (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
    END;

[DBG$NMATCH (.INPUT_DESC, DBG$CS_SYMBOLIZE, 2)]:
    BEGIN
    VERB_NODE [DBG$B_VERB_LITERAL] = SYMBOLIZE_VERB;
    DBG$GB_VERB = DBG$K_SYMBOLIZE_VERB;
    DBG$NPARSE_SYMBOLIZE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
```

```
3                   END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_TYPE, 1)]:
4               BEGIN
4               VERB_NODE [DBG$B_VERB_LITERAL] = TYPE_VERB;
4               DBG$GB_VERB = DBG$K_TYPE_VERB;
4               DBG$NPARSE_TYPE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3               END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_UNDEFINE, 1)]:
4               BEGIN
4               VERB_NODE [DBG$B_VERB_LITERAL] = UNDEFINE_VERB;
4               DBG$GB_VERB = DBG$K_UNDEFINE_VERB;
4               DBG$NPARSE_UNDEFINE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3               END;
3
3           [DBG$NMATCH (.INPUT_DESC, DBG$CS_WHILE, 1)]:
4               BEGIN
4               VERB_NODE [DBG$B_VERB_LITERAL] = WHILE_VERB;
4               DBG$GB_VERB = DBG$K_WHILE_VERB;
4               DBG$NPARSE_WHILE (.INPUT_DESC, .VERB_NODE, .MESSAGE_VECT)
3               END;
3
3
3           ! Any other command name constitutes a syntax error.
3           !
3           [OTHERWISE]:
4               BEGIN
4               .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC));
4               FALSE
3               END;
3
3           TES)
2       THEN
2           RETURN
2
2
2           ! Check for exhausted input
2           !
3           (IF .INPUT_DESC [DSC$W_LENGTH] EQL 0 OR
3               DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
3           THEN
3               STS$K_SUCCESS
3
3           ELSE
4               BEGIN
4               .MESSAGE_VECT = DBG$NSYNTAX_ERROR (DBG$NNEXT_WORD (.INPUT_DESC));
4               STS$K_SEVERE
4               END
3           )
2       ELSE
2           RETURN STS$K_SEVERE;
2
1       END;
```

.TITLE DBGNPARSE

```
                                                                .IDENT  \V04-000\

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

         45  54  41  43  4F  4C  4C  41  08  00000 P.AAA:       .ASCII  <8>\ALLOCATE\                                    ;
                                     40  01  00009 P.AAB:       .BYTE   1, 64                                            ;
                 48  43  41  54  54  41  06  0000B P.AAC:       .ASCII  <6>\ATTACH\                                      ;
                         4C  4C  41  43  04  00012 P.AAD:       .ASCII  <4>\CALL\                                        ;
                 4C  45  43  4E  41  43  06  00017 P.AAE:       .ASCII  <6>\CANCEL\                                      ;
             45  52  41  4C  43  45  44  07  0001E P.AAF:       .ASCII  <7>\DECLARE\                                     ;
                 45  4E  49  46  45  44  06  00026 P.AAG:       .ASCII  <6>\DEFINE\                                      ;
                 45  54  45  4C  45  44  06  0002D P.AAH:       .ASCII  <6>\DELETE\                                      ;
             54  49  53  4F  50  45  44  07  00034 P.AAI:       .ASCII  <7>\DEPOSIT\                                     ;
             59  41  4C  50  53  49  44  07  0003C P.AAJ:       .ASCII  <7>\DISPLAY\                                     ;
                         50  4D  55  44  04  00044 P.AAK:       .ASCII  <4>\DUMP\                                        ;
                         54  49  44  45  04  00049 P.AAL:       .ASCII  <4>\EDIT\                                        ;
         45  54  41  55  4C  41  56  45  08  0004E P.AAM:       .ASCII  <8>\EVALUATE\                                    ;
             45  4E  49  4D  41  58  45  07  00057 P.AAN:       .ASCII  <7>\EXAMINE\                                     ;
                         54  49  58  45  04  0005F P.AAO:       .ASCII  <4>\EXIT\                                        ;
         50  4F  4F  4C  54  49  58  45  08  00064 P.AAP:       .ASCII  <8>\EXITLOOP\                                    ;
                             52  4F  46  03  0006D P.AAQ:       .ASCII  <3>\FOR\                                         ;
                                 4F  47  02  00071 P.AAR:       .ASCII  <2>\GO\                                          ;
                         50  4C  45  48  04  00074 P.AAS:       .ASCII  <4>\HELP\                                        ;
                                 46  49  02  00079 P.AAT:       .ASCII  <2>\IF\                                          ;
                 54  41  45  50  45  52  06  0007C P.AAU:       .ASCII  <6>\REPEAT\                                      ;
                         45  56  41  53  04  00083 P.AAV:       .ASCII  <4>\SAVE\                                        ;
                 4C  4C  4F  52  43  53  06  00088 P.AAW:       .ASCII  <6>\SCROLL\                                      ;
                 48  43  52  41  45  53  06  0008F P.AAX:       .ASCII  <6>\SEARCH\                                      ;
                 54  43  45  4C  45  53  06  00096 P.AAY:       .ASCII  <6>\SELECT\                                      ;
                             54  45  53  03  0009D P.AAZ:       .ASCII  <3>\SET\                                         ;
                         57  4F  48  53  04  000A1 P.ABA:       .ASCII  <4>\SHOW\                                        ;
                     4E  57  41  50  53  05  000A6 P.ABB:       .ASCII  <5>\SPAWN\                                       ;
                         50  45  54  53  04  000AC P.ABC:       .ASCII  <4>\STEP\                                        ;
     45  5A  49  4C  4F  42  4D  59  53  09  000B1 P.ABD:       .ASCII  <9>\SYMBOLIZE\                                   ;
                         45  50  59  54  04  000BB P.ABE:       .ASCII  <4>\TYPE\                                        ;
         45  4E  49  46  45  44  4E  55  08  000C0 P.ABF:       .ASCII  <8>\UNDEFINE\                                    ;
                     45  4C  49  48  57  05  000C9 P.ABG:       .ASCII  <5>\WHILE\                                       ;
                                     0D  01  000CF P.ABH:       .BYTE   1, 13                                            ;

                                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                             00000 WORD_BUF:
                                                                .BLKB   81

                                                   DBG$CS_ALLOCATE=     P.AAA
                                                   DBG$CS_AT_SIGN=      P.AAB
                                                   DBG$CS_ATTACH=       P.AAC
                                                   DBG$CS_CALL=         P.AAD
                                                   DBG$CS_CANCEL=       P.AAE
                                                   DBG$CS_DECLARE=      P.AAF
                                                   DBG$CS_DEFINE=       P.AAG
                                                   DBG$CS_DELETE=       P.AAH
                                                   DBG$CS_DEPOSIT=      P.AAI
                                                   DBG$CS_DISPLAY=      P.AAJ
                                                   DBG$CS_DUMP=         P.AAK
                                                   DBG$CS_EDIT=         P.AAL
                                                   DBG$CS_EVALUATE=     P.AAM
```

```
DBG$CS_EXAMINE=         P.AAN
DBG$CS_EXIT=            P.AAO
DBG$CS_EXITLOOP=        P.AAP
DBG$CS_FOR=             P.AAQ
DBG$CS_GO=              P.AAR
DBG$CS_HELP=            P.AAS
DBG$CS_IF=              P.AAT
DBG$CS_REPEAT=          P.AAU
DBG$CS_SAVE=            P.AAV
DBG$CS_SCROLL=          P.AAW
DBG$CS_SEARCH=          P.AAX
DBG$CS_SELECT=          P.AAY
DBG$CS_SET=             P.AAZ
DBG$CS_SHOW=            P.ABA
DBG$CS_SPAWN=           P.ABB
DBG$CS_STEP=            P.ABC
DBG$CS_SYMBOLIZE=       P.ABD
DBG$CS_TYPE=            P.ABE
DBG$CS_UNDEFINE=        P.ABF
DBG$CS_WHILE=           P.ABG
DBG$CS_CR=              P.ABH
        .EXTRN  DBG$ADDR_EXP_INT
        .EXTRN  DBG$DEF_SYM_FIND
        .EXTRN  DBG$GET_MEMORY, DBG$GET_TEMPMEM
        .EXTRN  DBG$EXP_INT, DBG$NPARSE_ALLOCATE
        .EXTRN  DBG$NPARSE_AT_SIGN
        .EXTRN  DBG$NPARSE_ATTACH
        .EXTRN  DBG$NPARSE_CALL
        .EXTRN  DBG$NPARSE_CANCEL
        .EXTRN  DBG$NPARSE_DECLARE
        .EXTRN  DBG$NPARSE_DEFINE
        .EXTRN  DBG$NPARSE_DELETE
        .EXTRN  DBG$NPARSE_DEPOSIT
        .EXTRN  DBG$NPARSE_DUMP
        .EXTRN  DBG$NPARSE_EDIT
        .EXTRN  DBG$NPARSE_EVALUATE
        .EXTRN  DBG$NPARSE_EXAMINE
        .EXTRN  DBG$NPARSE_EXIT
        .EXTRN  DBG$NPARSE_EXITLOOP
        .EXTRN  DBG$NPARSE_FOR, DBG$NPARSE_GO
        .EXTRN  DBG$NPARSE_HELP
        .EXTRN  DBG$NPARSE_IF, DBG$NPARSE_REPEAT
        .EXTRN  DBG$NPARSE_SEARCH
        .EXTRN  DBG$NPARSE_SET, DBG$NPARSE_SHOW
        .EXTRN  DBG$NPARSE_SPAWN
        .EXTRN  DBG$NPARSE_STEP
        .EXTRN  DBG$NPARSE_SYMBOLIZE
        .EXTRN  DBG$NPARSE_TYPE
        .EXTRN  DBG$NPARSE_UNDEFINE
        .EXTRN  DBG$NPARSE_WHILE
        .EXTRN  DBG$NMAKE_ARG_VECT
        .EXTRN  DBG$NOUT_INFO, DBG$NREAD_NAME
        .EXTRN  DBG$NSET_LAST_TYPLEN
        .EXTRN  DBG$NSYNTAX_ERROR
        .EXTRN  DBG$REL_MEMORY, DBG$SCR_PARSE_DISPLAY_CMD
        .EXTRN  DBG$SCR_PARSE_SAVE_CMD
        .EXTRN  DBG$SCR_PARSE_SCROLL_CMD
```

```
                                                    .EXTRN  DBG$SCR_PARSE_SELECT_CMD
                                                    .EXTRN  DBG$STA_SETCONTEXT
                                                    .EXTRN  DBG$GL_DEVELOPER
                                                    .EXTRN  DBG$GL_EDIT_ENABLED
                                                    .EXTRN  DBG$GB_RADIX, DBG$GB_EXC_BRE_FLAG
                                                    .EXTRN  DBG$GB_LANGUAGE
                                                    .EXTRN  DBG$GL_SCREEN_NOGO
                                                    .EXTRN  DBG$GB_VERB

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

                                   01FC 00000       .ENTRY  DBG$NPARSE_CMD, Save R2,R3,R4,R5,R6,R7,R8     ; 0295
             58 00000000G   00       9E 00002       MOVAB   DBG$GL_SCREEN_NOGO, R8                        ;
             57 00000000G   00       9E 00009       MOVAB   LIB$SIGNAL, R7                                ;
             56     0000V   CF       9E 00010       MOVAB   DBG$NMATCH, R6                                ;
             55 00000000'   EF       9E 00015       MOVAB   DBG$CS_ALLOCATE, R5                           ;
             54 00000000G   00       9E 0001C       MOVAB   DBG$GB_VERB, R4                               ;
                               03    DD 00023       PUSHL   #3                                            ; 0383
    00000000G   00             01    FB 00025       CALLS   #1, DBG$GET_TEMPMEM                           ;
                53             50    D0 0002C       MOVL    R0, VERB_NODE                                 ;
           08   BC             53    D0 0002F       MOVL    VERB_NODE, @VERB_NODE_PTR                     ; 0384
                               64    94 00033       CLRB    DBG$GB_VERB                                   ; 0385
                               7E    D4 00035       CLRL    -(SP)                                         ; 0390
    00000000G   00             01    FB 00037       CALLS   #1, DBG$STA_SETCONTEXT                        ;
                               02    DD 0003E       PUSHL   #2                                            ; 0399
                               55    DD 00040       PUSHL   R5                                            ;
                52        04   AC    D0 00042       MOVL    INPUT_DESC, R2                                ;
                               52    DD 00046       PUSHL   R2                                            ;
                66             03    FB 00048       CALLS   #3, DBG$NMATCH                                ;
                01             50    D1 0004B       CMPL    R0, #1                                        ;
                               14    12 0004E       BNEQ    1$                                            ;
                63             16    90 00050       MOVB    #22, (VERB_NODE)                              ; 0401
                64             16    90 00053       MOVB    #22, DBG$GB_VERB                              ; 0402
                          0C   AC    DD 00056       PUSHL   MESSAGE_VECT                                  ; 0403
                               0C    BB 00059       PUSHR   #^M<R2,R3>                                    ;
    00000000G   00             03    FB 0005B       CALLS   #3, DBG$NPARSE_ALLOCATE                       ;
                               44    11 00062       BRB     3$                                            ;
                               01    DD 00064 1$:   PUSHL   #1                                            ; 0406
                          09   A5    9F 00066       PUSHAB  DBG$CS_AT_SIGN                                ;
                               52    DD 00069       PUSHL   R2                                            ;
                66             03    FB 0006B       CALLS   #3, DBG$NMATCH                                ;
                01             50    D1 0006E       CMPL    R0, #1                                        ;
                               14    12 00071       BNEQ    2$                                            ;
                63             01    90 00073       MOVB    #1, (VERB_NODE)                               ; 0408
                64             01    90 00076       MOVB    #1, DBG$GB_VERB                               ; 0409
                          0C   AC    DD 00079       PUSHL   MESSAGE_VECT                                  ; 0410
                               0C    BB 0007C       PUSHR   #^M<R2,R3>                                    ;
    00000000G   00             03    FB 0007E       CALLS   #3, DBG$NPARSE_AT_SIGN                        ;
                               60    11 00085       BRB     7$                                            ;
                               03    DD 00087 2$:   PUSHL   #3                                            ; 0413
                          0B   A5    9F 00089       PUSHAB  DBG$CS_ATTACH                                 ;
                               52    DD 0008C       PUSHL   R2                                            ;
                66             03    FB 0008E       CALLS   #3, DBG$NMATCH                                ;
                01             50    D1 00091       CMPL    R0, #1                                        ;
                               14    12 00094       BNEQ    4$                                            ;
                63             1A    90 00096       MOVB    #26, (VERB_NODE)                              ; 0415
                64             1A    90 00099       MOVB    #26, DBG$GB_VERB                              ; 0416
```

```
                          0C   AC  DD 0009C          PUSHL   MESSAGE_VECT                     : 0417
                               0C  BB 0009F          PUSHR   #^M<R2,R3>                       :
     00000000G  00             03  FB 000A1          CALLS   #3, DBG$NPARSE_ATTACH            :
                               60  11 000A8 3$:      BRB     9$                               :
                               03  DD 000AA 4$:      PUSHL   #3                               : 0429
                          12   A5  9F 000AC          PUSHAB  DBG$CS_CALL                      :
                               52  DD 000AF          PUSHL   R2                               :
                    66         03  FB 000B1          CALLS   #3, DBG$NMATCH                   :
                    01         50  D1 000B4          CMPL    R0, #1                           :
                               30  12 000B7          BNEQ    8$                               :
                    63         02  90 000B9          MOVB    #2, (VERB_NODE)                  : 0431
                    64         02  90 000BC          MOVB    #2, DBG$GB_VERB                  : 0432
                    09 00000000G   00  E9 000BF      BLBC    DBG$GB_EXC_BRE_FLAG, 5$          : 0433
                       00028E38    8F  DD 000C6      PUSHL   #167480                          :
                    67         01  FB 000CC          CALLS   #1, LIB$SIGNAL                   :
                    09         68  E9 000CF 5$:      BLBC    DBG$GL_SCREEN_NOGO, 6$           : 0434
                       0002836A    8F  DD 000D2      PUSHL   #164714                          :
                    67         01  FB 000D8          CALLS   #1, LIB$SIGNAL                   :
                          0C   AC  DD 000DB 6$:      PUSHL   MESSAGE_VECT                     : 0435
                               0C  BB 000DE          PUSHR   #^M<R2,R3>                       :
     00000000G  00             03  FB 000E0          CALLS   #3, DBG$NPARSE_CALL              :
                               67  11 000E7 7$:      BRB     12$                              :
                               03  DD 000E9 8$:      PUSHL   #3                               : 0438
                          17   A5  9F 000EB          PUSHAB  DBG$CS_CANCEL                    :
                               52  DD 000EE          PUSHL   R2                               :
                    66         03  FB 000F0          CALLS   #3, DBG$NMATCH                   :
                    01         50  D1 000F3          CMPL    R0, #1                           :
                               14  12 000F6          BNEQ    10$                              :
                    63         03  90 000F8          MOVB    #3, (VERB_NODE)                  : 0440
                    64         03  90 000FB          MOVB    #3, DBG$GB_VERB                  : 0441
                          0C   AC  DD 000FE          PUSHL   MESSAGE_VECT                     : 0442
                               0C  BB 00101          PUSHR   #^M<R2,R3>                       :
     00000000G  00             03  FB 00103          CALLS   #3, DBG$NPARSE_CANCEL            :
                               67  11 0010A 9$:      BRB     14$                              :
                               03  DD 0010C 10$:     PUSHL   #3                               : 0445
                          1E   A5  9F 0010E          PUSHAB  DBG$CS_DECLARE                   :
                               52  DD 00111          PUSHL   R2                               :
                    66         03  FB 00113          CALLS   #3, DBG$NMATCH                   :
                    01         50  D1 00116          CMPL    R0, #1                           :
                               14  12 00119          BNEQ    11$                              :
                    63         14  90 0011B          MOVB    #20, (VERB_NODE)                 : 0447
                    64         14  90 0011E          MOVB    #20, DBG$GB_VERB                 : 0448
                          0C   AC  DD 00121          PUSHL   MESSAGE_VECT                     : 0449
                               0C  BB 00124          PUSHR   #^M<R2,R3>                       :
     00000000G  00             03  FB 00126          CALLS   #3, DBG$NPARSE_DECLARE           :
                               67  11 0012D          BRB     16$                              :
                               03  DD 0012F 11$:     PUSHL   #3                               : 0452
                          26   A5  9F 00131          PUSHAB  DBG$CS_DEFINE                    :
                               52  DD 00134          PUSHL   R2                               :
                    66         03  FB 00136          CALLS   #3, DBG$NMATCH                   :
                    01         50  D1 00139          CMPL    R0, #1                           :
                               14  12 0013C          BNEQ    13$                              :
                    63         04  90 0013E          MOVB    #4, (VERB_NODE)                  : 0454
                    64         04  90 00141          MOVB    #4, DBG$GB_VERB                  : 0455
                          0C   AC  DD 00144          PUSHL   MESSAGE_VECT                     : 0456
                               0C  BB 00147          PUSHR   #^M<R2,R3>                       :
     00000000G  00             03  FB 00149          CALLS   #3, DBG$NPARSE_DEFINE            :
```

```
                         44  11 00150 12$:   BRB     16$                                 ;
                         03  DD 00152 13$:   PUSHL   #3                                  ; 0459
                    2D   A5  9F 00154        PUSHAB  DBG$CS_DELETE                       ;
                         52  DD 00157        PUSHL   R2                                  ;
              66         03  FB 00159        CALLS   #3, DBG$NMATCH                      ;
              01         50  D1 0015C        CMPL    R0, #1                              ;
                         14  12 0015F        BNEQ    15$                                 ;
              63         20  90 00161        MOVB    #32, (VERB_NODE)                    ; 0461
              64         20  90 00164        MOVB    #32, DBG$GB_VERB                    ; 0462
                    0C   AC  DD 00167        PUSHL   MESSAGE_VECT                        ; 0463
                         0C  BB 0016A        PUSHR   #^M<R2,R3>                          ;
    00000000G   00       03  FB 0016C        CALLS   #3, DBG$NPARSE_DELETE               ;
                         69  11 00173 14$:   BRB     19$                                 ;
                         01  DD 00175 15$:   PUSHL   #1                                  ; 0466
                    34   A5  9F 00177        PUSHAB  DBG$CS_DEPOSIT                      ;
                         52  DD 0017A        PUSHL   R2                                  ;
              66         03  FB 0017C        CALLS   #3, DBG$NMATCH                      ;
              01         50  D1 0017F        CMPL    R0, #1                              ;
                         14  12 00182        BNEQ    17$                                 ;
              63         05  90 00184        MOVB    #5, (VERB_NODE)                     ; 0468
              64         05  90 00187        MOVB    #5, DBG$GB_VERB                     ; 0469
                    0C   AC  DD 0018A        PUSHL   MESSAGE_VECT                        ; 0470
                         0C  BB 0018D        PUSHR   #^M<R2,R3>                          ;
    00000000G   00       03  FB 0018F        CALLS   #3, DBG$NPARSE_DEPOSIT              ;
                         7B  11 00196 16$:   BRB     24$                                 ;
                         03  DD 00198 17$:   PUSHL   #3                                  ; 0473
                    3C   A5  9F 0019A        PUSHAB  DBG$CS_DISPLAY                      ;
                         52  DD 0019D        PUSHL   R2                                  ;
              66         03  FB 0019F        CALLS   #3, DBG$NMATCH                      ;
              01         50  D1 001A2        CMPL    R0, #1                              ;
                         16  12 001A5        BNEQ    18$                                 ;
              63         1C  90 001A7        MOVB    #28, (VERB_NODE)                    ; 0475
              64         1C  90 001AA        MOVB    #28, DBG$GB_VERB                    ; 0476
                         53  DD 001AD        PUSHL   VERB_NODE                           ; 0477
                         7E  D4 001AF        CLRL    -(SP)                               ;
                         52  DD 001B1        PUSHL   R2                                  ;
    00000000G   00       03  FB 001B3        CALLS   #3, DBG$SCR_PARSE_DISPLAY_CMD       ;
                       0354  31 001BA        BRW     64$                                 ;
                         02  DD 001BD 18$:   PUSHL   #2                                  ; 0481
                    44   A5  9F 001BF        PUSHAB  DBG$CS_DUMP                         ;
                         52  DD 001C2        PUSHL   R2                                  ;
              66         03  FB 001C4        CALLS   #3, DBG$NMATCH                      ;
              01         50  D1 001C7        CMPL    R0, #1                              ;
                         14  12 001CA        BNEQ    20$                                 ;
              63         1B  90 001CC        MOVB    #27, (VERB_NODE)                    ; 0483
              64         1B  90 001CF        MOVB    #27, DBG$GB_VERB                    ; 0484
                    0C   AC  DD 001D2        PUSHL   MESSAGE_VECT                        ; 0485
                         0C  BB 001D5        PUSHR   #^M<R2,R3>                          ;
    00000000G   00       03  FB 001D7        CALLS   #3, DBG$NPARSE_DUMP                 ;
                         79  11 001DE 19$:   BRB     27$                                 ;
           07 00000000G  00  E8 001E0 20$:   BLBS    DBG$GL_EDIT_ENABLED, 21$            ; 0496
           0C 00000000G  00  E9 001E7        BLBC    DBG$GL_DEVELOPER, 22$               ;
                         02  DD 001EE 21$:   PUSHL   #2                                  ; 0498
                    49   A5  9F 001F0        PUSHAB  DBG$CS_EDIT                         ;
                         52  DD 001F3        PUSHL   R2                                  ;
              66         03  FB 001F5        CALLS   #3, DBG$NMATCH                      ;
                         02  11 001F8        BRB     23$                                 ;
```

```
                        50 D4 001FA 22$:    CLRL    R0                                   ; 0496
          01            50 D1 001FC 23$:    CMPL    R0, #1                               ;
                        14 12 001FF         BNEQ    25$                                  ;
          63            21 90 00201         MOVB    #33, (VERB_NODE)                     ; 0502
          64            21 90 00204         MOVB    #33, DBG$GB_VERB                     ; 0503
                  0C    AC DD 00207         PUSHL   MESSAGE_VECT                         ; 0504
                        0C BB 0020A         PUSHR   #^M<R2,R3>                           ;
00000000G 00            03 FB 0020C         CALLS   #3, DBG$NPARSE_EDIT                  ;
                        67 11 00213 24$:    BRB     29$                                  ;
                        02 DD 00215 25$:    PUSHL   #2                                   ; 0508
                  4E    A5 9F 00217         PUSHAB  DBG$CS_EVALUATE                      ;
                        52 DD 0021A         PUSHL   R2                                   ;
          66            03 FB 0021C         CALLS   #3, DBG$NMATCH                       ;
          01            50 D1 0021F         CMPL    R0, #1                               ;
                        14 12 00222         BNEQ    26$                                  ;
          63            06 90 00224         MOVB    #6, (VERB_NODE)                      ; 0510
          64            06 90 00227         MOVB    #6, DBG$GB_VERB                      ; 0511
                  0C    AC DD 0022A         PUSHL   MESSAGE_VECT                         ; 0512
                        0C BB 0022D         PUSHR   #^M<R2,R3>                           ;
00000000G 00            03 FB 0022F         CALLS   #3, DBG$NPARSE_EVALUATE              ;
                        67 11 00236         BRB     31$                                  ;
                        01 DD 00238 26$:    PUSHL   #1                                   ; 0515
                  57    A5 9F 0023A         PUSHAB  DBG$CS_EXAMINE                       ;
                        52 DD 0023D         PUSHL   R2                                   ;
          66            03 FB 0023F         CALLS   #3, DBG$NMATCH                       ;
          01            50 D1 00242         CMPL    R0, #1                               ;
                        14 12 00245         BNEQ    28$                                  ;
          63            07 90 00247         MOVB    #7, (VERB_NODE)                      ; 0517
          64            07 90 0024A         MOVB    #7, DBG$GB_VERB                      ; 0518
                  0C    AC DD 0024D         PUSHL   MESSAGE_VECT                         ; 0519
                        0C BB 00250         PUSHR   #^M<R2,R3>                           ;
00000000G 00            03 FB 00252         CALLS   #3, DBG$NPARSE_EXAMINE               ;
                        67 11 00259 27$:    BRB     33$                                  ;
                        03 DD 0025B 28$:    PUSHL   #3                                   ; 0522
                  5F    A5 9F 0025D         PUSHAB  DBG$CS_EXIT                          ;
                        52 DD 00260         PUSHL   R2                                   ;
          66            03 FB 00262         CALLS   #3, DBG$NMATCH                       ;
          01            50 D1 00265         CMPL    R0, #1                               ;
                        14 12 00268         BNEQ    30$                                  ;
          63            08 90 0026A         MOVB    #8, (VERB_NODE)                      ; 0524
          64            08 90 0026D         MOVB    #8, DBG$GB_VERB                      ; 0525
                  0C    AC DD 00270         PUSHL   MESSAGE_VECT                         ; 0526
                        0C BB 00273         PUSHR   #^M<R2,R3>                           ;
00000000G 00            03 FB 00275         CALLS   #3, DBG$NPARSE_EXIT                  ;
                        73 11 0027C 29$:    BRB     36$                                  ;
                        05 DD 0027E 30$:    PUSHL   #5                                   ; 0529
                  64    A5 9F 00280         PUSHAB  DBG$CS_EXITLOOP                      ;
                        52 DD 00283         PUSHL   R2                                   ;
          66            03 FB 00285         CALLS   #3, DBG$NMATCH                       ;
          01            50 D1 00288         CMPL    R0, #1                               ;
                        14 12 0028B         BNEQ    32$                                  ;
          63            13 90 0028D         MOVB    #19, (VERB_NODE)                     ; 0531
          64            13 90 00290         MOVB    #19, DBG$GB_VERB                     ; 0532
                  0C    AC DD 00293         PUSHL   MESSAGE_VECT                         ; 0533
                        0C BB 00296         PUSHR   #^M<R2,R3>                           ;
00000000G 00            03 FB 00298         CALLS   #3, DBG$NPARSE_EXITLOOP              ;
                        73 11 0029F 31$:    BRB     38$                                  ;
```

```
                              01 DD 002A1 32$:    PUSHL    #1                              ; 0536
                         6D   A5 9F 002A3         PUSHAB   DBG$CS_FOR                      ;
                              52 DD 002A6         PUSHL    R2                              ;
                   66         03 FB 002A8         CALLS    #3, DBG$NMATCH                  ;
                   01         50 D1 002AB         CMPL     R0, #1                          ;
                              14 12 002AE         BNEQ     34$                             ;
                   63         19 90 002B0         MOVB     #25, (VERB_NODE)                ; 0538
                   64         19 90 002B3         MOVB     #25, DBG$GB_VERB                ; 0539
                         0C   AC DD 002B6         PUSHL    MESSAGE_VECT                    ; 0540
                              0C BB 002B9         PUSHR    #^M<R2,R3>                      ;
        00000000G  00         03 FB 002BB         CALLS    #3, DBG$NPARSE_FOR              ;
                              73 11 002C2 33$:    BRB      40$                             ;
                              01 DD 002C4 34$:    PUSHL    #1                              ; 0543
                         71   A5 9F 002C6         PUSHAB   DBG$CS_GO                       ;
                              52 DD 002C9         PUSHL    R2                              ;
                   66         03 FB 002CB         CALLS    #3, DBG$NMATCH                  ;
                   01         50 D1 002CE         CMPL     R0, #1                          ;
                              20 12 002D1         BNEQ     37$                             ;
                   63         09 90 002D3         MOVB     #9, (VERB_NODE)                 ; 0545
                   64         09 90 002D6         MOVB     #9, DBG$GB_VERB                 ; 0546
                   09         68 E9 002D9         BLBC     DBG$GL_SCREEN_NOGO, 35$         ; 0547
                     0002836A 8F DD 002DC         PUSHL    #164714                         ;
                   67         01 FB 002E2         CALLS    #1, LIB$SIGNAL                  ;
                         0C   AC DD 002E5 35$:    PUSHL    MESSAGE_VECT                    ; 0548
                              0C BB 002E8         PUSHR    #^M<R2,R3>                      ;
        00000000G  00         03 FB 002EA         CALLS    #3, DBG$NPARSE_GO               ;
                              67 11 002F1 36$:    BRB      42$                             ;
                              01 DD 002F3 37$:    PUSHL    #1                              ; 0551
                         74   A5 9F 002F5         PUSHAB   DBG$CS_HELP                     ;
                              52 DD 002F8         PUSHL    R2                              ;
                   66         03 FB 002FA         CALLS    #3, DBG$NMATCH                  ;
                   01         50 D1 002FD         CMPL     R0, #1                          ;
                              14 12 00300         BNEQ     39$                             ;
                   63         0D 90 00302         MOVB     #13, (VERB_NODE)                ; 0553
                   64         0D 90 00305         MOVB     #13, DBG$GB_VERB                ; 0554
                         0C   AC DD 00308         PUSHL    MESSAGE_VECT                    ; 0555
                              0C BB 0030B         PUSHR    #^M<R2,R3>                      ;
        00000000G  00         03 FB 0030D         CALLS    #3, DBG$NPARSE_HELP             ;
                              44 11 00314 38$:    BRB      42$                             ;
                              01 DD 00316 39$:    PUSHL    #1                              ; 0558
                         79   A5 9F 00318         PUSHAB   DBG$CS_IF                       ;
                              52 DD 0031B         PUSHL    R2                              ;
                   66         03 FB 0031D         CALLS    #3, DBG$NMATCH                  ;
                   01         50 D1 00320         CMPL     R0, #1                          ;
                              14 12 00323         BNEQ     41$                             ;
                   63         10 90 00325         MOVB     #16, (VERB_NODE)                ; 0560
                   64         10 90 00328         MOVB     #16, DBG$GB_VERB                ; 0561
                         0C   AC DD 0032B         PUSHL    MESSAGE_VECT                    ; 0562
                              0C BB 0032E         PUSHR    #^M<R2,R3>                      ;
        00000000G  00         03 FB 00330         CALLS    #3, DBG$NPARSE_IF               ;
                              21 11 00337 40$:    BRB      42$                             ;
                              02 DD 00339 41$:    PUSHL    #2                              ; 0565
                         7C   A5 9F 0033B         PUSHAB   DBG$CS_REPEAT                   ;
                              52 DD 0033E         PUSHL    R2                              ;
                   66         03 FB 00340         CALLS    #3, DBG$NMATCH                  ;
                   01         50 D1 00343         CMPL     R0, #1                          ;
                              14 12 00346         BNEQ     43$                             ;
```

```
              63            12  90 00348        MOVB    #18, (VERB_NODE)                     ; 0567
              64            12  90 0034B        MOVB    #18, DBG$GB_VERB                     ; 0568
                      0C    AC  DD 0034E        PUSHL   MESSAGE_VECT                         ; 0569
                            0C  BB 00351        PUSHR   #^M<R2,R3>                           ;
    00000000G 00            03  FB 00353        CALLS   #3, DBG$NPARSE_REPEAT                ;
                            64  11 0035A  42$:  BRB     46$                                  ;
                            03  DD 0035C  43$:  PUSHL   #3                                   ; 0572
                    0083    C5  9F 0035E        PUSHAB  DBG$CS_SAVE                          ;
                            52  DD 00362        PUSHL   R2                                   ;
              66            03  FB 00364        CALLS   #3, DBG$NMATCH                       ;
              01            50  D1 00367        CMPL    R0, #1                               ;
                            11  12 0036A        BNEQ    44$                                  ;
              63            1F  90 0036C        MOVB    #31, (VERB_NODE)                     ; 0574
              64            1F  90 0036F        MOVB    #31, DBG$GB_VERB                     ; 0575
                            0C  BB 00372        PUSHR   #^M<R2,R3>                           ; 0576
    00000000G 00            02  FB 00374        CALLS   #2, DBG$SCR_PARSE_SAVE_CMD           ;
                            64  11 0037B        BRB     48$                                  ;
                            03  DD 0037D  44$:  PUSHL   #3                                   ; 0580
                    0088    C5  9F 0037F        PUSHAB  DBG$CS_SCROLL                        ;
                            52  DD 00383        PUSHL   R2                                   ;
              66            03  FB 00385        CALLS   #3, DBG$NMATCH                       ;
              01            50  D1 00388        CMPL    R0, #1                               ;
                            11  12 0038B        BNEQ    45$                                  ;
              63            1D  90 0038D        MOVB    #29, (VERB_NODE)                     ; 0582
              64            1D  90 00390        MOVB    #29, DBG$GB_VERB                     ; 0583
                            0C  BB 00393        PUSHR   #^M<R2,R3>                           ; 0584
    00000000G 00            02  FB 00395        CALLS   #2, DBG$SCR_PARSE_SCROLL_CMD         ;
                            43  11 0039C        BRB     48$                                  ;
                            03  DD 0039E  45$:  PUSHL   #3                                   ; 0588
                    008F    C5  9F 003A0        PUSHAB  DBG$CS_SEARCH                        ;
                            52  DD 003A4        PUSHL   R2                                   ;
              66            03  FB 003A6        CALLS   #3, DBG$NMATCH                       ;
              01            50  D1 003A9        CMPL    R0, #1                               ;
                            14  12 003AC        BNEQ    47$                                  ;
              63            0F  90 003AE        MOVB    #15, (VERB_NODE)                     ; 0590
              64            0F  90 003B1        MOVB    #15, DBG$GB_VERB                     ; 0591
                      0C    AC  DD 003B4        PUSHL   MESSAGE_VECT                         ; 0592
                            0C  BB 003B7        PUSHR   #^M<R2,R3>                           ;
    00000000G 00            03  FB 003B9        CALLS   #3, DBG$NPARSE_SEARCH                ;
                            68  11 003C0  46$:  BRB     51$                                  ;
                            03  DD 003C2  47$:  PUSHL   #3                                   ; 0595
                    0096    C5  9F 003C4        PUSHAB  DBG$CS_SELECT                        ;
                            52  DD 003C8        PUSHL   R2                                   ;
              66            03  FB 003CA        CALLS   #3, DBG$NMATCH                       ;
              01            50  D1 003CD        CMPL    R0, #1                               ;
                            12  12 003D0        BNEQ    49$                                  ;
              63            1E  90 003D2        MOVB    #30, (VERB_NODE)                     ; 0597
              64            1E  90 003D5        MOVB    #30, DBG$GB_VERB                     ; 0598
                            0C  BB 003D8        PUSHR   #^M<R2,R3>                           ; 0599
    00000000G 00            02  FB 003DA        CALLS   #2, DBG$SCR_PARSE_SELECT_CMD         ;
                          012D  31 003E1  48$:  BRW     64$                                  ;
                            02  DD 003E4  49$:  PUSHL   #2                                   ; 0603
                    009D    C5  9F 003E6        PUSHAB  DBG$CS_SET                           ;
                            52  DD 003EA        PUSHL   R2                                   ;
              66            03  FB 003EC        CALLS   #3, DBG$NMATCH                       ;
              01            50  D1 003EF        CMPL    R0, #1                               ;
                            14  12 003F2        BNEQ    50$                                  ;
```

```
                 63            0A  90 003F4        MOVB    #10, (VERB_NODE)                          ; 0605
                 64            0A  90 003F7        MOVB    #10, DBG$GB_VERB                          ; 0606
                          0C   AC  DD 003FA        PUSHL   MESSAGE_VECT                              ; 0607
                               0C  BB 003FD        PUSHR   #^M<R2,R3>
       00000000G  00           03  FB 003FF        CALLS   #3, DBG$NPARSE_SET
                               76  11 00406        BRB     55$
                               02  DD 00408 50$:   PUSHL   #2                                        ; 0610
                        00A1   C5  9F 0040A        PUSHAB  DBG$CS_SHOW
                               52  DD 0040E        PUSHL   R2
                 66            03  FB 00410        CALLS   #3, DBG$NMATCH
                 01            50  D1 00413        CMPL    R0, #1
                               14  12 00416        BNEQ    52$
                 63            0B  90 00418        MOVB    #11, (VERB_NODE)                          ; 0612
                 64            0B  90 0041B        MOVB    #11, DBG$GB_VERB                          ; 0613
                          0C   AC  DD 0041E        PUSHL   MESSAGE_VECT                              ; 0614
                               0C  BB 00421        PUSHR   #^M<R2,R3>
       00000000G  00           03  FB 00423        CALLS   #3, DBG$NPARSE_SHOW
                               76  11 0042A 51$:   BRB     57$
                               02  DD 0042C 52$:   PUSHL   #2                                        ; 0617
                        00A6   C5  9F 0042E        PUSHAB  DBG$CS_SPAWN
                               52  DD 00432        PUSHL   R2
                 66            03  FB 00434        CALLS   #3, DBG$NMATCH
                 01            50  D1 00437        CMPL    R0, #1
                               14  12 0043A        BNEQ    53$
                 63            15  90 0043C        MOVB    #21, (VERB_NODE)                          ; 0619
                 64            15  90 0043F        MOVB    #21, DBG$GB_VERB                          ; 0620
                          0C   AC  DD 00442        PUSHL   MESSAGE_VECT                              ; 0621
                               0C  BB 00445        PUSHR   #^M<R2,R3>
       00000000G  00           03  FB 00447        CALLS   #3, DBG$NPARSE_SPAWN
                               76  11 0044E        BRB     59$
                               01  DD 00450 53$:   PUSHL   #1                                        ; 0624
                        00AC   C5  9F 00452        PUSHAB  DBG$CS_STEP
                               52  DD 00456        PUSHL   R2
                 66            03  FB 00458        CALLS   #3, DBG$NMATCH
                 01            50  D1 0045B        CMPL    R0, #1
                               20  12 0045E        BNEQ    56$
                 63            0C  90 00460        MOVB    #12, (VERB_NODE)                          ; 0626
                 64            0C  90 00463        MOVB    #12, DBG$GB_VERB                          ; 0627
                 09            68  E9 00466        BLBC    DBG$GL_SCREEN_NOGO, 54$                   ; 0628
                    0002836A   8F  DD 00469        PUSHL   #164714
                 67            01  FB 0046F        CALLS   #1, LIB$SIGNAL
                          0C   AC  DD 00472 54$:   PUSHL   MESSAGE_VECT                              ; 0629
                               0C  BB 00475        PUSHR   #^M<R2,R3>
       00000000G  00           03  FB 00477        CALLS   #3, DBG$NPARSE_STEP
                               6A  11 0047E 55$:   BRB     61$
                               02  DD 00480 56$:   PUSHL   #2                                        ; 0632
                        00B1   C5  9F 00482        PUSHAB  DBG$CS_SYMBOLIZE
                               52  DD 00486        PUSHL   R2
                 66            03  FB 00488        CALLS   #3, DBG$NMATCH
                 01            50  D1 0048B        CMPL    R0, #1
                               14  12 0048E        BNEQ    58$
                 63            17  90 00490        MOVB    #23, (VERB_NODE)                          ; 0634
                 64            17  90 00493        MOVB    #23, DBG$GB_VERB                          ; 0635
                          0C   AC  DD 00496        PUSHL   MESSAGE_VECT                              ; 0636
                               0C  BB 00499        PUSHR   #^M<R2,R3>
       00000000G  00           03  FB 0049B        CALLS   #3, DBG$NPARSE_SYMBOLIZE
                               6A  11 004A2 57$:   BRB     63$                                       ;
```

```
                                 01    DD 004A4 58$:   PUSHL    #1                                  ; 0639
                      00BB       C5    9F 004A6        PUSHAB   DBG$CS_TYPE                         ;
                                 52    DD 004AA        PUSHL    R2                                  ;
              66                 03    FB 004AC        CALLS    #3, DBG$NMATCH                      ;
              01                 50    D1 004AF        CMPL     R0, #1                              ;
                                 14    12 004B2        BNEQ     60$                                 ;
              63                 0E    90 004B4        MOVB     #14, (VERB_NODE)                    ; 0641
              64                 0E    90 004B7        MOVB     #14, DBG$GB_VERB                    ; 0642
                         0C      AC    DD 004BA        PUSHL    MESSAGE_VECT                        ; 0643
                                 0C    BB 004BD        PUSHR    #^M<R2,R3>                          ;
       00000000G    00           03    FB 004BF        CALLS    #3, DBG$NPARSE_TYPE                 ;
                                 46    11 004C6 59$:   BRB      63$                                 ;
                                 01    DD 004C8 60$:   PUSHL    #1                                  ; 0646
                      00C0       C5    9F 004CA        PUSHAB   DBG$CS_UNDEFINE                     ;
                                 52    DD 004CE        PUSHL    R2                                  ;
              66                 03    FB 004D0        CALLS    #3, DBG$NMATCH                      ;
              01                 50    D1 004D3        CMPL     R0, #1                              ;
                                 14    12 004D6        BNEQ     62$                                 ;
              63                 18    90 004D8        MOVB     #24, (VERB_NODE)                    ; 0648
              64                 18    90 004DB        MOVB     #24, DBG$GB_VERB                    ; 0649
                         0C      AC    DD 004DE        PUSHL    MESSAGE_VECT                        ; 0650
                                 0C    BB 004E1        PUSHR    #^M<R2,R3>                          ;
       00000000G    00           03    FB 004E3        CALLS    #3, DBG$NPARSE_UNDEFINE             ;
                                 22    11 004EA 61$:   BRB      63$                                 ;
                                 01    DD 004EC 62$:   PUSHL    #1                                  ; 0653
                      00C9       C5    9F 004EE        PUSHAB   DBG$CS_WHILE                        ;
                                 52    DD 004F2        PUSHL    R2                                  ;
              66                 03    FB 004F4        CALLS    #3, DBG$NMATCH                      ;
              01                 50    D1 004F7        CMPL     R0, #1                              ;
                                 2B    12 004FA        BNEQ     66$                                 ;
              63                 11    90 004FC        MOVB     #17, (VERB_NODE)                    ; 0655
              64                 11    90 004FF        MOVB     #17, DBG$GB_VERB                    ; 0656
                         0C      AC    DD 00502        PUSHL    MESSAGE_VECT                        ; 0657
                                 0C    BB 00505        PUSHR    #^M<R2,R3>                          ;
       00000000G    00           03    FB 00507        CALLS    #3, DBG$NPARSE_WHILE                ;
                    2A           50    E9 0050E 63$:   BLBC     R0, 67$                             ;
                                 62    B5 00511 64$:   TSTW     (R2)                                ; 0676
                                 0E    13 00513        BEQL     65$                                 ;
                                 01    DD 00515        PUSHL    #1                                  ; 0677
                      00CF       C5    9F 00517        PUSHAB   DBG$CS_CR                           ;
                                 52    DD 0051B        PUSHL    R2                                  ;
              66                 03    FB 0051D        CALLS    #3, DBG$NMATCH                      ;
              04                 50    E9 00520        BLBC     R0, 66$                             ;
              50                 01    D0 00523 65$:   MOVL     #1, R0                              ; 0676
                                       04 00526        RET                                          ;
                                 52    DD 00527 66$:   PUSHL    R2                                  ; 0683
           0000V    CF           01    FB 00529        CALLS    #1, DBG$NNEXT_WORD                  ;
                                 50    DD 0052E        PUSHL    R0                                  ;
       00000000G    00           01    FB 00530        CALLS    #1, DBG$NSYNTAX_ERROR               ;
              0C    BC           50    D0 00537        MOVL     R0, @MESSAGE_VECT                   ;
              50                 04    D0 0053B 67$:   MOVL     #4, R0                              ; 0688
                                       04 0053E        RET                                          ; 0690

; Routine Size:  1343 bytes,    Routine Base:  DBG$CODE + 0000
```

```
GLOBAL ROUTINE DBG$NMATCH (STRING_DESC, COUNTED_STRING, UNIQUE_CHARS) =
!
! FUNCTION
!       This routine is used extensively during command parsing. What it does is
!       to compare the first word of the command input string against the supplied
!       counted string to see if they match. If they do, the matched word is removed
!       from the head of the command input string and success is returned as the value
!       of the routine. Note that the input word may be shorter than the counted
!       string to allow for abbreviations of commands.
!
! FORMAL PARAMETERS:
!
!       STRING_DESC      - VAX standard string descriptor of the input command.
!
!       COUNTED_STRING   - The counted string against which the input word is
!                          to be compared.
!
!       UNIQUE_CHARS     - The number of characters which the input word must
!                          match against the counted string for purposes of
!                          disambiguating the input word.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       Unsigned integer longword.
!
! COMPLETION CODES:
!
!       STS$K_SEVERE (4)  - The input word did not match the counted string.
!
!       STS$K_SUCCESS (1) - The input word did match.
!
! SIDE EFFECTS:
!
!       On a successful match, the input string descriptor is updated to just beyond
!       the word of input matched against the counted string.
!

    BEGIN

    MAP
        STRING_DESC : REF DBG$STG_DESC,
        COUNTED_STRING: REF VECTOR[,BYTE];

    LOCAL
        WORD_STRING: REF VECTOR[,BYTE], ! Holds the next input word
        SAVE_PTR,                       ! Saves the address of the input buffer
        SAVE_LEN,                       ! Saves the length of the input buffer
        MATCH_FLAG;                     ! Signals a match

```

```


    ! Save the original descriptor buffer address and length
    !
    SAVE_PTR = .STRING_DESC [DSC$A_POINTER];
    SAVE_LEN = .STRING_DESC [DSC$W_LENGTH];


    ! Get the next input word
    !
    WORD_STRING = DBG$NNEXT_WORD (.STRING_DESC);


    ! Perform the match, check for special cases
    !
    MATCH_FLAG = TRUE;
    SELECTONE TRUE OF
        SET


        ! Match
        !
        [.COUNTED_STRING [0] EQL 0]:
            0;


        ! No match
        !
        [.WORD_STRING [0] GTR .COUNTED_STRING [0]]:
            MATCH_FLAG = FALSE;


        ! No match
        !
        [.WORD_STRING [0] EQL 0]:
            MATCH_FLAG = FALSE;


        ! No match
        !
        [.WORD_STRING [0] LSS .UNIQUE_CHARS]:
            MATCH_FLAG = FALSE;


        ! Normal match
        !
        [OTHERWISE]:
            BEGIN

            LOCAL
                I;                              ! Loop counter

            I = 1;
            WHILE .I LEQ .WORD_STRING [0] DO
                BEGIN
                IF .WORD_STRING [.I] NEQ .COUNTED_STRING [.I]
                THEN
```

```
;  675          0805  4                   MATCH_FLAG = FALSE;
;  676          0806  4
;  677          0807  4               I = .I + 1;
;  678          0808  3               END;
;  679          0809  3
;  680          0810  2           END;
;  681          0811  2
;  682          0812  2       TES;
;  683          0813  2
;  684          0814  2
;  685          0815  2   ! Return successfully if we found a match.  Otherwise, restore the old
;  686          0816  2   ! descriptor and return failure to find a match.
;  687          0817  2   !
;  688          0818  2   IF .MATCH_FLAG THEN RETURN STS$K_SUCCESS;
;  689          0819  2   STRING_DESC [DSC$A_POINTER] = .SAVE_PTR;
;  690          0820  2   STRING_DESC [DSC$W_LENGTH] = .SAVE_LEN;
;  691          0821  2   RETURN STS$K_SEVERE;
;  692          0822  2
;  693          0823  1   END;



                                          003C 00000        .ENTRY  DBG$NMATCH, Save R2,R3,R4,R5               ; 0691
                        52       04   AC    D0 00002        MOVL    STRING_DESC, R2                            ; 0752
                        55       04   A2    D0 00006        MOVL    4(R2), SAVE_PTR                            ;
                        54            62    3C 0000A        MOVZWL  (R2), SAVE_LEN                             ; 0753
                                      52    DD 0000D        PUSHL   R2                                         ; 0758
                 0000V  CF            01    FB 0000F        CALLS   #1, DBG$NNEXT_WORD                         ;
                        53            01    D0 00014        MOVL    #1, MATCH_FLAG                             ; 0763
                                 08   BC    95 00017        TSTB    @COUNTED_STRING                            ; 0770
                                      2E    13 0001A        BEQL    5$                                         ;
                    08  BC            60    91 0001C        CMPB    (WORD_STRING), @COUNTED_STRING             ; 0776
                                      0C    1A 00020        BGTRU   1$                                         ;
                                      60    95 00022        TSTB    (WORD_STRING)                              ; 0782
                                      08    13 00024        BEQL    1$                                         ;
 0C    AC           60          08    00    ED 00026        CMPZV   #0, #8, (WORD_STRING), UNIQUE_CHARS        ; 0788
                                      04    18 0002C        BGEQ    2$                                         ;
                                      53    D4 0002E 1$:    CLRL    MATCH_FLAG                                 ; 0789
                                      18    11 00030        BRB     5$                                         ;
                        51            01    D0 00032 2$:    MOVL    #1, I                                      ; 0800
       51           60          08    00    ED 00035 3$:    CMPZV   #0, #8, (WORD_STRING), I                   ; 0801
                                      0E    19 0003A        BLSS    5$                                         ;
                    08 BC41         6140    91 0003C        CMPB    (I)[WORD_STRING], @COUNTED_STRING[I]       ; 0803
                                      02    13 00042        BEQL    4$                                         ;
                                      53    D4 00044        CLRL    MATCH_FLAG                                 ; 0805
                                      51    D6 00046 4$:    INCL    I                                          ; 0807
                                      EB    11 00048        BRB     3$                                         ; 0801
                        04            53    E9 0004A 5$:    BLBC    MATCH_FLAG, 6$                             ; 0818
                        50            01    D0 0004D        MOVL    #1, R0                                     ;
                                      04       00050        RET                                                ;
                    04  A2            55    D0 00051 6$:    MOVL    SAVE_PTR, 4(R2)                            ; 0819
                        62            54    B0 00055        MOVW    SAVE_LEN, (R2)                             ; 0820
                        50            04    D0 00058        MOVL    #4, R0                                     ; 0821
                                      04       0005B        RET                                                ; 0823
```

; Routine Size: 92 bytes,   Routine Base: DBG$CODE + 053F

```
GLOBAL ROUTINE DBG$NNEXT_WORD (STRING_DESC) =
!
! FUNCTION
!       Routine DBG$NNEXT_WORD isolates the next word of the input command. A word
!       is defined to be any number of alphabetic characters delimited by blanks
!       or a single non-alphabetic character. The word isolated is represented as
!       a counted string.
!
!       The input string is assumed to be terminated by a <cr>.
!
!       Maximum word size is 80 characters.
!
! FORMAL PARAMETERS:
!
!       STRING_DESC     - VAX standard string descriptor of the input command.
!
! IMPLICIT INPUTS:
!
!       WORD_BUF        - OWNed byte vector to contain the word of input.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       The address of WORD_BUF, the counted string representing the next word
!       of input.
!
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       WORD_BUF is always filled with the next word of input. WORD_BUF[0]
!       contains the count of the characters in the word.  On exhausted input,
!       WORD_BUF[0] equals 0.  The command string descriptor is updated past
!       the word of input.
!

    BEGIN

    MAP
        STRING_DESC : REF DBG$STG_DESC; ! Input string descriptor

    LOCAL
        CHAR : BYTE,                    ! Holds next character of input
        POINTER,                        ! Temp pointer
        COUNT;                          ! Count of characters


     ! Check for exhausted input
     !
     IF .STRING_DESC [DSC$W_LENGTH] LEQ 0
     THEN
```

```
3           BEGIN
3           STRING_DESC [DSC$W_LENGTH] = 0;
3           WORD_BUF [0] = 0;                  ! No word
3           STRING_DESC [DSC$A_POINTER] = 0;
3           RETURN WORD_BUF [0];
2           END;
2
2
2       ! Ignore leading white space
2       !
2       WHILE CH$RCHAR (.STRING_DESC [DSC$A_POINTER]) EQL ' ' DO
3           BEGIN
3           STRING_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_DESC [DSC$A_POINTER], 1);
3           STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
2           END;
2
2
2       ! Count the number of characters in the next word.  Note that we always
2       ! return at least one character.
2       !
2       POINTER = .STRING_DESC [DSC$A_POINTER];
2       CHAR = CH$RCHAR (.STRING_DESC [DSC$A_POINTER]);
3       IF (.CHAR GEQ 'A' AND .CHAR LEQ 'Z')
2       THEN
3           BEGIN
3
3
3           ! We take more than one char
3           !
3           CHAR = CH$A_RCHAR (STRING_DESC [DSC$A_POINTER]);
3           STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
3           WHILE ((.CHAR GEQ 'A' AND .CHAR LEQ 'Z') OR .CHAR EQL '_') AND
3                 CH$DIFF (.STRING_DESC [DSC$A_POINTER], .POINTER) LEQU WORD_SIZE
3           DO
4               BEGIN
4               CHAR = CH$A_RCHAR (STRING_DESC [DSC$A_POINTER]);
4               STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
3               END;
3
3           END
3
3
3       ! Take one character only.
3       !
2       ELSE
3           BEGIN
3           CHAR = CH$A_RCHAR (STRING_DESC [DSC$A_POINTER]);
3           STRING_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH] - 1;
2           END;
2
2
2       ! Calculate the number of characters in the new word
2       !
2       COUNT = CH$DIFF (.STRING_DESC [DSC$A_POINTER], .POINTER);
2
2
2       ! Now copy over th appropriate number of chars.
```

```
2   !
2   WORD_BUF [0] = .COUNT;
2   CH$MOVE (.COUNT, .POINTER, WORD_BUF [1]);
2
2
2   ! Check for exhausted input
2   !
2   IF .STRING_DESC [DSC$W_LENGTH] LEQ 0
2   THEN
3       BEGIN
3       STRING_DESC [DSC$W_LENGTH] = 0;
3       STRING_DESC [DSC$A_POINTER] = 0;
2       END;
2
2   RETURN WORD_BUF[0];
2
1   END;
```

```
                                01FC 00000        .ENTRY  DBG$NNEXT_WORD, Save R2,R3,R4,R5,R6,R7,R8    ; 0824
          58 00000000'  EF  9E 00002        MOVAB   WORD_BUF, R8                                   ;
          57        04  AC  D0 00009        MOVL    STRING_DESC, R7                                ; 0879
                        67  B5 0000D        TSTW    (R7)                                           ;
                        09  12 0000F        BNEQ    1$                                             ;
                        67  B4 00011        CLRW    (R7)                                           ; 0882
                        68  94 00013        CLRB    WORD_BUF                                       ; 0883
                    04  A7  D4 00015        CLRL    4(R7)                                          ; 0884
                        68  11 00018        BRB     9$                                             ; 0885
          56        04  A7  9E 0001A 1$:    MOVAB   4(R7), R6                                      ; 0891
          20        00  B6  91 0001E 2$:    CMPB    @0(R6), #32                                    ;
                        06  12 00022        BNEQ    3$                                             ;
                        66  D6 00024        INCL    (R6)                                           ; 0893
                        67  B7 00026        DECW    (R7)                                           ; 0894
                        F4  11 00028        BRB     2$                                             ; 0891
          52            66  D0 0002A 3$:    MOVL    (R6), POINTER                                  ; 0901
          50        00  B6  90 0002D        MOVB    @0(R6), CHAR                                   ; 0902
      41  8F            50  91 00031        CMPB    CHAR, #65                                      ; 0903
                        2F  1F 00035        BLSSU   7$                                             ;
      5A  8F            50  91 00037        CMPB    CHAR, #90                                      ;
                        29  1A 0003B        BGTRU   7$                                             ;
                        66  D6 0003D 4$:    INCL    (R6)                                           ; 0910
          50        00  B6  90 0003F        MOVB    @0(R6), CHAR                                   ;
                        67  B7 00043        DECW    (R7)                                           ; 0911
      41  8F            50  91 00045        CMPB    CHAR, #65                                      ; 0912
                        06  1F 00049        BLSSU   5$                                             ;
      5A  8F            50  91 0004B        CMPB    CHAR, #90                                      ;
                        06  1B 0004F        BLEQU   6$                                             ;
      5F  8F            50  91 00051 5$:    CMPB    CHAR, #95                                      ;
                        17  12 00055        BNEQ    8$                                             ;
  51            66      52  C3 00057 6$:    SUBL3   POINTER, (R6), R1                              ; 0913
   00000050     8F      51  D1 0005B        CMPL    R1, #80                                        ;
                        0A  1A 00062        BGTRU   8$                                             ;
                        D7  11 00064        BRB     4$                                             ; 0916
                        66  D6 00066 7$:    INCL    (R6)                                           ; 0927
```

```
                   50      00    B6  90 00068           MOVB    @0(R6), CHAR                        ;
                                 67  B7 0006C           DECW    (R7)                                ; 0928
          50       66            52  C3 0006E 8$:       SUBL3   POINTER, (R6), COUNT                ; 0934
                   68            50  90 00072           MOVB    COUNT, WORD_BUF                     ; 0939
     01   A8       62            50  28 00075           MOVC3   COUNT, (POINTER), WORD_BUF+1        ; 0940
                                 67  B5 0007A           TSTW    (R7)                                ; 0945
                                 04  12 0007C           BNEQ    9$                                  ;
                                 67  B4 0007E           CLRW    (R7)                                ; 0948
                                 66  D4 00080           CLRL    (R6)                                ; 0949
                   50            68  9E 00082 9$:       MOVAB   WORD_BUF, R0                        ; 0952
                                     04 00085           RET                                         ; 0954

; Routine Size:  134 bytes,    Routine Base:  DBG$CODE + 059B
```

```
GLOBAL ROUTINE DBG$NPARSE_EXPRESSION(INPUT_DESC, RADIX,
                                VALUE_DESC_PTR, TERM_INDEX, MESSAGE_VECT) =
!
! FUNCTION
!       This routine interfaces to the Expression Interpreter.
!       It first checks whether the expression is a DEFINEd name, in
!       which case the expansion is done.
!
! FORMAL PARAMETERS:
!
!       INPUT_DESC      - The address of a VAX Standard ASCII string descriptor
!                         which describes the user input.
!
!       RADIX           - A longword containing an integer encoding of the
!                         radix to be used for the interpretation of numeric
!                         literals.
!
!       VALUE_DESC_PTR  - The address of a longword to contain the address
!                         of a language specific value descriptor.
!
!       TERM_INDEX      - The "terminator index" value which indicates which
!                         lexical tokens are valid expression terminators in
!                         the current context.  For example, in the IF command,
!                         the keyword "THEN" is the valid expression termina-
!                         tor.  These index values have names of the form
!                         TOKEN$K_TERM_xxx.
!
!       MESSAGE_VECT    - The address of a longword to contain the address
!                         of a message argument vector for errors.
!
!       6th Optional Parameter - If this is present, and the value is
!                         DBG$K_DEPOSIT_VERB then pass this into DBG$EXP_INT,
!                         and into DBG$EXPRESSION_PARSER, so that in
!                         DBG$EXPRESSION_PARSER, when the expression is not
!                         address expression and in deposit command,
!                         DBG$EVAL_LANG_OPERATOR will not be called with
!                         DBG$GL_IDENTITY_TOKEN.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       Same as the Expression Interpreter.
!
! COMPLETION CODES:
!
!       Same as the Expression Interpreter.
!
! SIDE EFFECTS:
!
!       Same as the Expression Interpreter.
```

```
;  884       1012  1 !
;  885       1013  1
;  886       1014  2      BEGIN
;  887       1015  2
;  888       1016  2      BUILTIN ACTUALCOUNT,ACTUALPARAMETER;
;  889       1017  2
;  890       1018  2
;  891       1019  2      ! Translate default radix.
;  892       1020  2      !
;  893       1021  2      IF .RADIX EQL DBG$K_DEFAULT
;  894       1022  2      THEN
;  895       1023  2          RADIX = .DBG$GB_RADIX[DBG$B_RADIX_INPUT];
;  896       1024  2
;  897       1025  2      IF ACTUALCOUNT() GTR 5
;  898       1026  2      THEN
;  899       1027  3          BEGIN
;  900       1028  3          IF ACTUALPARAMETER(6) NEQ DBG$K_DEPOSIT_VERB
;  901       1029  3          THEN
;  902       1030  3              $DBG_ERROR('DBGNPARSE\DBG$NPARSE_EXPRESSION');
;  903       1031  3
;  904       1032  3          RETURN DBG$EXP_INT (.INPUT_DESC, .RADIX, .VALUE_DESC_PTR, .TERM_INDEX,
;  905       1033  3                  DBG$K_DEPOSIT_VERB);
;  906       1034  3          END
;  907       1035  3
;  908       1036  2      ELSE
;  909       1037  2          RETURN DBG$EXP_INT (.INPUT_DESC, .RADIX, .VALUE_DESC_PTR, .TERM_INDEX);
;  910       1038  1      END;


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

24  47  42  44  5C  45  53  52  41  50  4E  47  42  44  1F  000D1 P.ABI:  .ASCII  <31>\DBGNPARSE\<92>\DBG$NPARSE_EXPRESS\     ;
    53  53  45  52  50  58  45  5F  45  53  52  41  50  4E  000E0                                                             ;
                                                4E  4F  49  000EE         .ASCII  \ION\                                        ;


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

                                                    0004  00000         .ENTRY  DBG$NPARSE_EXPRESSION, Save R2           ; 0955
                          52  0000000G   00    9E  00002                MOVAB   DBG$EXP_INT, R2                          ;
                          01         08  AC    D1  00009                CMPL    RADIX, #1                                ; 1021
                                         08    12  0000D                BNEQ    1$                                       ;
                     08   AC  0000000G   00    9A  0000F                MOVZBL  DBG$GB_RADIX, RADIX                      ; 1023
                          05             6C    91  00017  1$:           CMPB    (AP), #5                                 ; 1025
                                         29    1B  0001A                BLEQU   3$                                       ;
                          05         18  AC    D1  0001C                CMPL    24(AP), #5                               ; 1028
                                         15    13  00020                BEQL    2$                                       ;
                              00000000'  EF    9F  00022                PUSHAB  P.ABI                                    ; 1030
                                         01    DD  00028                PUSHL   #1                                       ;
                              00028362   8F    DD  0002A                PUSHL   #164706                                  ;
               0000000G   00             03    FB  00030                CALLS   #3, LIB$SIGNAL                           ;
                                         05    DD  00037  2$:           PUSHL   #5                                       ; 1032
                          7E         0C  AC    7D  00039                MOVQ    VALUE_DESC_PTR, -(SP)                    ;
                          7E         04  AC    7D  0003D                MOVQ    INPUT_DESC, -(SP)                        ;
                          62             05    FB  00041                CALLS   #5, DBG$EXP_INT                          ;
```

```
                          04 00044          RET                                   ; 1037
7E        0C    AC    7D 00045 3$:       MOVQ    VALUE_DESC_PTR, -(SP)           ;
7E        04    AC    7D 00049           MOVQ    INPUT_DESC, -(SP)               ;
62              04    FB 0004D           CALLS   #4, DBG$EXP_INT                 ;
                          04 00050          RET                                   ; 1038
```

; Routine Size: 81 bytes,   Routine Base: DBG$CODE + 0621

```
GLOBAL ROUTINE DBG$NPARSE_ADDRESS(INPUT_DESC, ADDR_EXP_PTR,
                                  RADIX, TERM_INDEX, MESSAGE_VECT) =
!
! FUNCTION
!       This routine interfaces to the Address Expression Interpreter to obtain
!       an address expression descriptor. Address expression descriptors are
!       used to describe the object of commands which use address expressions.
!       The descriptor may contain a pointer to a primary descriptor, a pointer
!       to a permanent symbol descriptor, an untyped L-value, or an L-value of
!       type instruction.
!
! INPUTS
!       INPUT_DESC      - A longword containing the address of a standard
!                         string descriptor describing the input command.
!
!       ADDR_EXP_PTR    - The address of a longword to contain the address
!                         of an address expression descriptor.
!
!       RADIX           - A longword containing the integer code of the
!                         radix to be used in interpreting numeric literals.
!
!       TERM_INDEX      - The "terminator index" value which indicates which
!                         lexical tokens are valid expression terminators in
!                         the current context.  For example, in the EXAMINE
!                         command, comma and colon are valid terminators (as
!                         is carriage-return).  These index values have names
!                         of the form TOKEN$K_TERM_xxx.
!
!       MESSAGE_VECT    - The address of a longword to contain the address
!                         of a message argument vector for errors.
!
! OUTPUTS
!       INPUT_DESC      - The input string descriptor is updated to point to
!                         the first character after the address expression
!                         that was just parsed.
!
!       ADDR_EXP_PTR    - An Address Expression Descriptor is constructed and
!                         its address is returned to ADDR_EXP_PTR.
!
!       MESSAGE_VECT    - If an error is encountered, a message argument vector
!                         may be constructed (unless the error is singalled)
!                         and its address returned to MESSAGE_VECT.
!
!       An unsigned integer longword completion code is returned as the routine
!                         value.  This is whatever completion code the Address
!                         Expression Interpreter returns.
!
    BEGIN

    MAP
        INPUT_DESC: REF DBG$STG_DESC;

    BIND
        DBG$CS_CR = UPLIT BYTE (1, DBG$K_CAR_RETURN);

    LOCAL
        LENGTH,                         ! Current location length
```

```
;   969        1096  2              STATUS,                             ! Return status
;   970        1097  2              TYPE;                               ! Current location type
;   971        1098  2
;   972        1099  2
;   973        1100  2
;   974        1101  2       ! Initialize type and length
;   975        1102  2       !
;   976        1103  2       TYPE = DBG$K_NOTYPE;
;   977        1104  2       LENGTH = 0;
;   978        1105  2
;   979        1106  2
;   980        1107  2       ! Translate default radix.
;   981        1108  2       !
;   982        1109  2       IF .RADIX EQL DBG$K_DEFAULT
;   983        1110  2       THEN
;   984        1111  2           RADIX = .DBG$GB_RADIX[DBG$B_RADIX_INPUT];
;   985        1112  2
;   986        1113  2
;   987        1114  2       ! Now call the Address Expression Interpreter appropriate for the current
;   988        1115  2       ! language.
;   989        1116  2       !
;   990        1117  2       STATUS = DBG$ADDR_EXP_INT(.INPUT_DESC, .ADDR_EXP_PTR,
;   991        1118  2                                 TYPE, LENGTH, .RADIX, .TERM_INDEX);
;   992        1119  2
;   993        1120  2       ! Set the potential type and length for current location.
;   994        1121  2       !
;   995        1122  2       DBG$NSET_LAST_TYPLEN (.TYPE, .LENGTH);
;   996        1123  2       RETURN .STATUS;
;   997        1124  2
;   998        1125  1       END;


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

                            0D  01  000F1 P.ABJ:    .BYTE   1, 13                                    ;

                                          DBG$CS_CR=              P.ABJ

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

                                0004 00000          .ENTRY  DBG$NPARSE_ADDRESS, Save R2              ; 1039
          7E         80      8F   9A 00002          MOVZBL  #128, TYPE                               ; 1103
                             7E   D4 00006          CLRL    LENGTH                                   ; 1104
          01         0C      AC   D1 00008          CMPL    RADIX, #1                                ; 1109
                             08   12 0000C          BNEQ    1$                                       ;
     0C   AC  00000000G      00   9A 0000E          MOVZBL  DBG$GB_RADIX, RADIX                      ; 1111
          7E         0C      AC   7D 00016 1$:      MOVQ    RADIX, -(SP)                             ; 1118
                     08      AE   9F 0001A          PUSHAB  LENGTH                                   ; 1117
                     10      AE   9F 0001D          PUSHAB  TYPE                                     ;
          7E         04      AC   7D 00020          MOVQ    INPUT_DESC, -(SP)                        ;
00000000G 00                 06   FB 00024          CALLS   #6, DBG$ADDR_EXP_INT                     ;
          52                 50   D0 0002B          MOVL    R0, STATUS                               ;
                             6E   DD 0002E          PUSHL   LENGTH                                   ; 1122
                     08      AE   DD 00030          PUSHL   TYPE                                     ;
00000000G 00                 02   FB 00033          CALLS   #2, DBG$NSET_LAST_TYPLEN                 ;
```

```
                50            52  D0 0003A        MOVL    STATUS, R0                    ; 1123
                                  04 0003D        RET                                   ; 1125

; Routine Size:  62 bytes,    Routine Base:  DBG$CODE + 0672
```

```
1 GLOBAL ROUTINE DBG$NSAVE_DECIMAL_INTEGER(INPUT_DESC, RESULT) =
1 !
1 ! FUNCTION
1 !       Extracts a numeric string from the command input string and converts it
1 !       to a binary longword integer.  Decimal radix is assumed for the input.
1 !
1 ! INPUTS
1 !       INPUT_DESC - The address of a standard ASCII string descriptor
1 !                    pointing to the input command line to be scanned.
1 !
1 !       RESULT  - The address of a longword to receive the result, meaning
1 !                 the scanned integer value in internal format.
1 !
1 ! OUTPUTS
1 !       INPUT_DESC - The input string descriptor is updated to point to the
1 !                    first character after the scanned integer constant.
1 !
1 !       RESULT  - The scanned integer value (in internal format) is returned
1 !                 the RESULT location.
1 !
1 !       The value STS$K_SUCCESS is always returned as the routine value.
1 !
1
2     BEGIN
2
2     BUILTIN
2         EMUL;                               ! Multiply and add two longwords
2                                             !         to produce quadword result.
2
2     MAP
2         INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor
2
2     BIND
2         DBG$CS_CR = UPLIT BYTE(1, DBG$K_CAR_RETURN);
2
2     LOCAL
2         VALUE: VECTOR[2],                   ! Result quadword
2         STRING_PTR,                         ! Pointer to input string
2         TRUNC_FLAG,                         ! Indicates truncation
2         CHAR;                               ! Holds characters
2
2
2
2     ! Check for null input
2     !
2     IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
2     THEN
2         SIGNAL(DBG$_NEEDMORE);
2
2
2     ! Delete leading white space
2     !
2     STRING_PTR = .INPUT_DESC [DSC$A_POINTER];
2     WHILE .INPUT_DESC [DSC$W_LENGTH] GTR 0 DO
3         BEGIN
3         CHAR = CH$RCHAR_A (STRING_PTR);
3         INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
```

```
        IF (.CHAR NEQ DBG$K_BLANK) AND (.CHAR NEQ DBG$K_TAB) THEN EXITLOOP;
        END;


    ! Check for a numeric character. If we don't have one - syntax error.
    !
    IF (.CHAR LSS '0') OR (.CHAR GTR '9')
    THEN
        BEGIN
        INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
        DBG$SYNTAX_ERROR(.INPUT_DESC);
        END;


    ! Now continue to accept numeric chars and convert them.  Loop until we
    ! find a non-numeric character.  In the loop, append each new digit to
    ! the number being accumulated, check for overflow, and loop for the next
    ! character.
    !
    VALUE [0] = 0;
    VALUE [1] = 0;
    TRUNC_FLAG = FALSE;
    WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '9') DO
        BEGIN
        CHAR = .CHAR - '0';
        EMUL (VALUE [0], %REF (10), CHAR, VALUE);
        IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
        INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
        CHAR = CH$RCHAR_A (STRING_PTR);
        END;


    ! If the next character is an alphabetic character, signal a syntax error.
    ! This can happen if the user tries to enter a hex number, for example.
    !
    IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z')
    THEN
        SIGNAL(DBG$_INVDIGDEC, 2, 1, CHAR);


    ! Update the input pointer and length. Remember that char right now contains
    ! the first non-numeric character of the remaining string. Therefore, string_ptr
    ! points one place to the right too far.
    !
    INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] + 1;
    INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);


    ! Set up the result, check for truncation, and check for a negative number.
    !
    IF .TRUNC_FLAG THEN SIGNAL(DBG$_NUMTRUNC);
    IF .VALUE [0] LSS 0 THEN VALUE [0] = -.VALUE [0];
    .RESULT = .VALUE [0];
    RETURN STS$K_SUCCESS;

    END;
```

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

                          0D  01  000F3 P.ABK:    .BYTE   1, 13                                          ;

                                        DBG$CS_CR=            P.ABK

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

                                003C  00000       .ENTRY  DBG$NSAVE_DECIMAL_INTEGER, Save R2,R3,R4,R5  ; 1126
               55 00000000G   00   9E  00002       MOVAB   LIB$SIGNAL, R5                               ;
               5E             0C   C2  00009       SUBL2   #12, SP                                      ;
                              01   DD  0000C       PUSHL   #1                                           ; 1171
                 00000000'    EF   9F  0000E       PUSHAB  DBG$CS_CR                                    ;
               52        04   AC   D0  00014       MOVL    INPUT_DESC, R2                               ;
                              52   DD  00018       PUSHL   R2                                           ;
          FE70  CF            03   FB  0001A       CALLS   #3, DBG$NMATCH                               ;
                09            50   E9  0001F       BLBC    R0, 1$                                       ;
                 000280D0     8F   DD  00022       PUSHL   #164048                                      ; 1173
               65             01   FB  00028       CALLS   #1, LIB$SIGNAL                               ;
               54        04   A2   D0  0002B 1$:   MOVL    4(R2), STRING_PTR                            ; 1178
                              62   B5  0002F 2$:   TSTW    (R2)                                         ; 1179
                              0F   13  00031       BEQL    3$                                           ;
               6E             84   9A  00033       MOVZBL  (STRING_PTR)+, CHAR                          ; 1181
                              62   B7  00036       DECW    (R2)                                         ; 1182
               20             6E   D1  00038       CMPL    CHAR, #32                                    ; 1183
                              F2   13  0003B       BEQL    2$                                           ;
               09             6E   D1  0003D       CMPL    CHAR, #9                                     ;
                              ED   13  00040       BEQL    2$                                           ;
               30             6E   D1  00042 3$:   CMPL    CHAR, #48                                    ; 1189
                              05   19  00045       BLSS    4$                                           ;
               39             6E   D1  00047       CMPL    CHAR, #57                                    ;
                              0C   15  0004A       BLEQ    5$                                           ;
          04   A2        FF   A4   9E  0004C 4$:   MOVAB   -1(R4), 4(R2)                                ; 1192
                              52   DD  00051       PUSHL   R2                                           ; 1193
         0000V  CF            01   FB  00053       CALLS   #1, DBG$SYNTAX_ERROR                         ;
                         04   AE   7C  00058 5$:   CLRQ    VALUE                                        ; 1202
                              53   D4  0005B       CLRL    TRUNC_FLAG                                   ; 1204
               30             6E   D1  0005D 6$:   CMPL    CHAR, #48                                    ; 1205
                              1E   19  00060       BLSS    8$                                           ;
               39             6E   D1  00062       CMPL    CHAR, #57                                    ;
                              19   14  00065       BGTR    8$                                           ;
               6E             30   C2  00067       SUBL2   #48, CHAR                                    ; 1207
  04   AE        6E           0A   04   AE   7A  0006A       EMUL    VALUE, #10, CHAR, VALUE            ; 1208
                         08   AE   D5  00071       TSTL    VALUE+4                                      ; 1209
                              03   13  00074       BEQL    7$                                           ;
               53             01   D0  00076       MOVL    #1, TRUNC_FLAG                               ;
                              62   B7  00079 7$:   DECW    (R2)                                         ; 1210
               6E             84   9A  0007B       MOVZBL  (STRING_PTR)+, CHAR                          ; 1211
                              DD   11  0007E       BRB     6$                                           ; 1205
       00000041   8F          6E   D1  00080 8$:   CMPL    CHAR, #65                                    ; 1218
                              18   19  00087       BLSS    9$                                           ;
       0000005A   8F          6E   D1  00089       CMPL    CHAR, #90                                    ;
                              0F   14  00090       BGTR    9$                                           ;
                              5E   DD  00092       PUSHL   SP                                           ; 1220
```

```
                         01   DD 00094        PUSHL   #1                     ;
                         02   DD 00096        PUSHL   #2                     ;
            00028AAA     8F   DD 00098        PUSHL   #166570                ;
       65                04   FB 0009E        CALLS   #4, LIB$SIGNAL         ;
                         62   B6 000A1 9$:    INCW    (R2)                   ; 1227
 04    A2          FF    A4   9E 000A3        MOVAB   -1(R4), 4(R2)          ; 1228
       09                53   E9 000A8        BLBC    TRUNC_FLAG, 10$        ; 1233
            00028043     8F   DD 000AB        PUSHL   #163907                ;
       65                01   FB 000B1        CALLS   #1, LIB$SIGNAL         ;
                   04    AE   D5 000B4 10$:   TSTL    VALUE                  ; 1234
                         05   18 000B7        BGEQ    11$                    ;
 04    AE          04    AE   CE 000B9        MNEGL   VALUE, VALUE           ;
 08    BC          04    AE   D0 000BE 11$:   MOVL    VALUE, @RESULT         ; 1235
       50                01   D0 000C3        MOVL    #1, R0                 ; 1236
                              04 000C6        RET                            ; 1238
```

; Routine Size: 199 bytes,   Routine Base: DBG$CODE + 06h0

```
GLOBAL ROUTINE DBG$NSAVE_INTEGER(INPUT_DESC, RESULT) =
!
! FUNCTION
!       Extracts a numeric string from the command input string and converts it
!       to a binary longword integer.  The default radix is assumed for the
!       input.
!
! INPUTS
!       INPUT_DESC - The address of a standard ASCII string descriptor
!                    pointing to the input command line to be scanned.
!
!       RESULT  - The address of a longword to receive the result, meaning
!                    the scanned integer value in internal format.
!
! OUTPUTS
!       INPUT_DESC - The input string descriptor is updated to point to the
!                    first character after the scanned integer constant.
!
!       RESULT  - The scanned integer value (in internal format) is returned
!                    the RESULT location.
!
!       The value STS$K_SUCCESS is always returned as the routine value.
!

    BEGIN

    BUILTIN
        EMUL;                               ! Multiply and add two longwords
                                            !       to produce quadword result.

    MAP
        INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor

    BIND
        DBG$CS_CR = UPLIT BYTE(1, DBG$K_CAR_RETURN);

    LOCAL
        VALUE: VECTOR[2],                   ! Result quadword
        STRING_PTR,                         ! Pointer to input string
        TRUNC_FLAG,                         ! Indicates truncation
        CHAR;                               ! Holds characters


    ! Check for null input
    !
    IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
    THEN
        SIGNAL(DBG$_NEEDMORE);


    ! Delete leading white space
    !
    STRING_PTR = .INPUT_DESC [DSC$A_POINTER];
    WHILE .INPUT_DESC [DSC$W_LENGTH] GTR 0 DO
        BEGIN
        CHAR = CH$RCHAR_A (STRING_PTR);
```

```
        INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
        IF (.CHAR NEQ DBG$K_BLANK) AND (.CHAR NEQ DBG$K_TAB) THEN EXITLOOP;
        END;

    ! Case on the current radix setting.  We will believe this number to
    ! be in that radix.
    !
    SELECTONE .DBG$GB_RADIX[DBG$B_RADIX_INPUT] OF
        SET

        [DBG$K_BINARY] :
            BEGIN

            ! Check for a binary character. If we don't have one - syntax error.
            !
            IF (.CHAR LSS '0') OR (.CHAR GTR '1')
            THEN
                BEGIN
                INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
                DBG$SYNTAX_ERROR(.INPUT_DESC);
            END;

            ! Now continue to accept binary chars and convert them.  Loop
            ! until we find a non-numeric character.  In the loop, append
            ! each new digit to the number being accumulated, check for
            ! overflow, and loop for the next character.
            !
            VALUE [0] = 0;
            VALUE [1] = 0;
            TRUNC_FLAG = FALSE;
            WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '1') DO
            BEGIN
                CHAR = .CHAR - '0';
                EMUL (VALUE [0], %REF (2), CHAR, VALUE);
                IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
                INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
                CHAR = CH$RCHAR_A (STRING_PTR);
            END;

            ! If the next character is an alphabetic character, signal a
            ! syntax error.  This can happen if the user tries to enter a
            ! hex number, for example.
            !
            IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z') OR
               (.CHAR GEQ '2') AND (.CHAR LEQ '9')
            THEN
                SIGNAL(DBG$_INVDIGBIN, 2, 1, CHAR);

            END;

        [DBG$K_OCTAL] :
            BEGIN

            ! Check for a octal character. If we don't have one - syntax error.
            !
            IF (.CHAR LSS '0') OR (.CHAR GTR '7')
            THEN
```

```
4                   BEGIN
4                   INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
4                   DBG$SYNTAX_ERROR(.INPUT_DESC);
3               END;
3
3               ! Now continue to accept numeric chars and convert them.  Loop
3               ! until we find a non-numeric character.  In the loop, append
3               ! each new digit to the number being accumulated, check for
3               ! overflow, and loop for the next character.
3               !
3               VALUE [0] = 0;
3               VALUE [1] = 0;
3               TRUNC_FLAG = FALSE;
3               WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '7') DO
4               BEGIN
4                   CHAR = .CHAR - '0';
4                   EMUL (VALUE [0], %REF (8), CHAR, VALUE);
4                   IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
4                   INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
4                   CHAR = CH$RCHAR_A (STRING_PTR);
3               END;
3
3               ! If the next character is an alphabetic character, signal a
3               ! syntax error.  This can happen if the user tries to enter a
3               ! hex number, for example.
3               !
3               IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z') OR
4                  (.CHAR GEQ '8') AND (.CHAR LEQ '9')
3               THEN
3                   SIGNAL(DBG$_INVDIGOCT, 2, 1, CHAR);
3
2               END;
2
2           [DBG$K_DECIMAL] :
3               BEGIN
3
3               ! Check for a decimal character. If we don't have one - syntax error.
3               !
4               IF (.CHAR LSS '0') OR (.CHAR GTR '9')
3               THEN
4                   BEGIN
4                   INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
4                   DBG$SYNTAX_ERROR(.INPUT_DESC);
3               END;
3
3               ! Now continue to accept numeric chars and convert them.  Loop
3               ! until we find a non-numeric character.  In the loop, append
3               ! each new digit to the number being accumulated, check for
3               ! overflow, and loop for the next character.
3               !
3               VALUE [0] = 0;
3               VALUE [1] = 0;
3               TRUNC_FLAG = FALSE;
3               WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '9') DO
4               BEGIN
4                   CHAR = .CHAR - '0';
4                   EMUL (VALUE [0], %REF (10), CHAR, VALUE);
```

```
        IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
        INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
        CHAR = CH$RCHAR_A (STRING_PTR);
    END;

    ! If the next character is an alphabetic character, signal a
    ! syntax error.  This can happen if the user tries to enter a
    ! hex number, for example.
    !
    IF (.CHAR GEQ 'A') AND (.CHAR LEQ 'Z')
    THEN
        SIGNAL(DBG$_INVDIGDEC, 2, 1, CHAR);

    END;

[DBG$K_HEX] :
    BEGIN

    ! Check for a hexidecimal character. If we don't have one - syntax error.
    !
    IF ((.CHAR LSS '0') OR (.CHAR GTR '9')) AND
       ((.CHAR LSS 'A') OR (.CHAR GTR 'F'))
    THEN
        BEGIN
        INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
        DBG$SYNTAX_ERROR(.INPUT_DESC);
    END;

    ! Now continue to accept hexidecimal chars and convert them.
    ! Loop until we find a non-numeric character.  In the loop,
    ! append each new digit to the number being accumulated, check
    ! for overflow, and loop for the next character.
    !
    VALUE [0] = 0;
    VALUE [1] = 0;
    TRUNC_FLAG = FALSE;
    WHILE (.CHAR GEQ '0') AND (.CHAR LEQ '9') OR
          (.CHAR GEQ 'A') AND (.CHAR LEQ 'F') DO
    BEGIN
        IF (.CHAR GEQ '0') AND (.CHAR LEQ '9')
        THEN
            CHAR = .CHAR - '0'
        ELSE
            CHAR = .CHAR - 55;
        EMUL (VALUE [0], %REF (16), CHAR, VALUE);
        IF .VALUE [1] NEQ 0 THEN TRUNC_FLAG = TRUE;
        INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
        CHAR = CH$RCHAR_A (STRING_PTR);
    END;

    ! If the next character is an alphabetic character, signal a
    ! syntax error.
    !
    IF (.CHAR GEQ 'G') AND (.CHAR LEQ 'Z')
    THEN
        SIGNAL(DBG$_INVDIGHEX, 2, 1, CHAR);

```

```
; 1342          1467  2             END;
; 1343          1468  2
; 1344          1469  2         [OTHERWISE] :
; 1345          1470  3             BEGIN
; 1346          1471  3             $DBG_ERROR('DBGNPARSE\DBG$NSAVE_INTEGER, DBG$GB_RADIX is invalid');
; 1347          1472  3             END;
; 1348          1473  2
; 1349          1474  2         TES;
; 1350          1475  2
; 1351          1476  2     ! Update the input pointer and length. Remember that char right now contains
; 1352          1477  2     ! the first non-numeric character of the remaining string. Therefore, string_ptr
; 1353          1478  2     ! points one place to the right too far.
; 1354          1479  2     !
; 1355          1480  2     INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] + 1;
; 1356          1481  2     INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.STRING_PTR, -1);
; 1357          1482  2
; 1358          1483  2
; 1359          1484  2     ! Set up the result, check for truncation, and check for a negative number.
; 1360          1485  2     !
; 1361          1486  2     IF .TRUNC_FLAG THEN SIGNAL(DBG$_NUMTRUNC);
; 1362          1487  2     IF .VALUE [0] LSS 0 THEN VALUE [0] = -.VALUE [0];
; 1363          1488  2     .RESULT = .VALUE [0];
; 1364          1489  2     RETURN STS$K_SUCCESS;
; 1365          1490  2
; 1366          1491  1     END;


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

                                                 0D  01  000F5 P.ABL:  .BYTE   1, 13
24 47 42 44 5C 45 53 52 41 50 4E 47 42 44 34  000F7 P.ABM:  .ASCII  \4DBGNPARSE\<92>\DBG$NSAVE_INTEGER, DBG$\      ;
20 2C 52 45 47 45 54 4E 49 5F 45 56 41 53 4E  00106                                                             ;
                                     24 47 42 44  00115                                                             ;
76 6E 69 20 73 69 20 58 49 44 41 52 5F 42 47  00119         .ASCII  \GB_RADIX is invalid\                       ;
                                     64 69 6C 61  00128                                                             ;

                                                   DBG$CS_CR=            P.ABL


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

                                            00FC 00000          .ENTRY  DBG$NSAVE_INTEGER, Save R2,R3,R4,R5,R6,R7   ; 1239
                     57        0000V  CF  9E 00002          MOVAB   DBG$SYNTAX_ERROR, R7                         ;
                     56   00000000G  00  9E 00007          MOVAB   LIB$SIGNAL, R6                               ;
                     5E               0C  C2 0000E          SUBL2   #12, SP                                      ;
                                      01  DD 00011          PUSHL   #1                                           ; 1285
                          00000000'  EF  9F 00013          PUSHAB  DBG$CS_CR                                    ;
                     52          04  AC  D0 00019          MOVL    INPUT_DESC, R2                               ;
                                      52  DD 0001D          PUSHL   R2                                           ;
               FDA4  CF               03  FB 0001F          CALLS   #3, DBG$NMATCH                               ;
                     09               50  E9 00024          BLBC    R0, 1$                                       ;
                          000280D0   8F  DD 00027          PUSHL   #164048                                      ; 1287
                     66               01  FB 0002D          CALLS   #1, LIB$SIGNAL                               ;
                     53          04  A2  9E 00030 1$:      MOVAB   4(R2), R3                                    ; 1292
                     54               63  D0 00034          MOVL    (R3), STRING_PTR                             ;
                                      62  B5 00037 2$:      TSTW    (R2)                                         ; 1293
```

```
                                    0F  13 00039            BEQL    3$                              ;
                              6E    84  9A 0003B            MOVZBL  (STRING_PTR)+, CHAR             ; 1295
                                    62  B7 0003E            DECW    (R2)                            ; 1296
                              20    6E  D1 00040            CMPL    CHAR, #32                       ; 1297
                                    F2  13 00043            BEQL    2$                              ;
                              09    6E  D1 00045            CMPL    CHAR, #9                        ;
                                    ED  13 00048            BEQL    2$                              ;
                              50 00000000G  00  9A 0004A 3$:  MOVZBL  DBG$GB_RADIX, R0              ; 1303
                              02    50  91 00051            CMPB    R0, #2                          ; 1306
                                    65  12 00054            BNEQ    11$                             ;
                              30    6E  D1 00056            CMPL    CHAR, #48                       ; 1311
                                    05  19 00059            BLSS    4$                              ;
                              31    6E  D1 0005B            CMPL    CHAR, #49                       ;
                                    09  15 0005E            BLEQ    5$                              ;
                              63    FF  A4  9E 00060 4$:    MOVAB   -1(R4), (R3)                    ; 1314
                                    52  DD 00064            PUSHL   R2                              ; 1315
                              67    01  FB 00066            CALLS   #1, DBG$SYNTAX_ERROR            ;
                              04    AE  7C 00069 5$:        CLRQ    VALUE                           ; 1323
                                    55  D4 0006C            CLRL    TRUNC_FLAG                      ; 1325
                              30    6E  D1 0006E 6$:        CMPL    CHAR, #48                       ; 1326
                                    1E  19 00071            BLSS    8$                              ;
                              31    6E  D1 00073            CMPL    CHAR, #49                       ;
                                    19  14 00076            BGTR    8$                              ;
                              6E    30  C2 00078            SUBL2   #48, CHAR                       ; 1328
04    AE            6E        02    04  AE  7A 0007B        EMUL    VALUE, #2, CHAR, VALUE          ; 1329
                              08    AE  D5 00082            TSTL    VALUE+4                         ; 1330
                                    03  13 00085            BEQL    7$                              ;
                              55    01  D0 00087            MOVL    #1, TRUNC_FLAG                  ;
                              62    B7 0008A 7$:            DECW    (R2)                            ; 1331
                              6E    84  9A 0008C            MOVZBL  (STRING_PTR)+, CHAR             ; 1332
                                    DD  11 0008F            BRB     6$                              ; 1326
              00000041        8F    6E  D1 00091 8$:        CMPL    CHAR, #65                       ; 1339
                                    09  19 00098            BLSS    9$                              ;
              0000005A        8F    6E  D1 0009A            CMPL    CHAR, #90                       ;
                                    0A  15 000A1            BLEQ    10$                             ;
                              32    6E  D1 000A3 9$:        CMPL    CHAR, #50                       ; 1340
                                    68  19 000A6            BLSS    18$                             ;
                              39    6E  D1 000A8            CMPL    CHAR, #57                       ;
                                    68  14 000AB            BGTR    19$                             ;
                                    5E  DD 000AD 10$:       PUSHL   SP                              ; 1342
                                    01  DD 000AF            PUSHL   #1                              ;
                                    02  DD 000B1            PUSHL   #2                              ;
                   00028AB2         8F  DD 000B3            PUSHL   #166578                         ;
                                    68  11 000B9            BRB     21$                             ;
                              08    50  91 000BB 11$:       CMPB    R0, #8                          ; 1346
                                    65  12 000BE            BNEQ    22$                             ;
                              30    6E  D1 000C0            CMPL    CHAR, #48                       ; 1351
                                    05  19 000C3            BLSS    12$                             ;
                              37    6E  D1 000C5            CMPL    CHAR, #55                       ;
                                    09  15 000C8            BLEQ    13$                             ;
                              63    FF  A4  9E 000CA 12$:   MOVAB   -1(R4), (R3)                    ; 1354
                                    52  DD 000CE            PUSHL   R2                              ; 1355
                              67    01  FB 000D0            CALLS   #1, DBG$SYNTAX_ERROR            ;
                              04    AE  7C 000D3 13$:       CLRQ    VALUE                           ; 1363
                                    55  D4 000D6            CLRL    TRUNC_FLAG                      ; 1365
                              30    6E  D1 000D8 14$:       CMPL    CHAR, #48                       ; 1366
                                    1E  19 000DB            BLSS    16$                             ;
```

```
                                   37         6E  D1 000DD         CMPL     CHAR, #55                          :
                                              19  14 000E0         BGTR     16$                                :
                                   6E         30  C2 000E2         SUBL2    #48, CHAR                          : 1368
   04   AE        6E               08    04   AE  7A 000E5         EMUL     VALUE, #8, CHAR, VALUE             : 1369
                                         08   AE  D5 000EC         TSTL     VALUE+4                            : 1370
                                              03  13 000EF         BEQL     15$                                :
                                   55         01  D0 000F1         MOVL     #1, TRUNC_FLAG                     :
                                              62  B7 000F4  15$:   DECW     (R2)                               : 1371
                                   6E         84  9A 000F6         MOVZBL   (STRING_PTR)+, CHAR                : 1372
                                              DD  11 000F9         BRB      14$                                : 1366
                       00000041    8F         6E  D1 000FB  16$:   CMPL     CHAR, #65                          : 1379
                                              09  19 00102         BLSS     17$                                :
                       0000005A    8F         6E  D1 00104         CMPL     CHAR, #90                          :
                                              0A  15 0010B         BLEQ     20$                                :
                                   38         6E  D1 0010D  17$:   CMPL     CHAR, #56                          : 1380
                                              5A  19 00110  18$:   BLSS     28$                                :
                                   39         6E  D1 00112         CMPL     CHAR, #57                          :
                                              61  14 00115  19$:   BGTR     31$                                :
                                              5E  DD 00117  20$:   PUSHL    SP                                 : 1382
                                              01  DD 00119         PUSHL    #1                                 :
                                              02  DD 0011B         PUSHL    #2                                 :
                                   00028AC2   8F  DD 0011D         PUSHL    #166594                            :
                                              61  11 00123  21$:   BRB      32$                                :
                                   0A         50  91 00125  22$:   CMPB     R0, #10                            : 1386
                                              5F  12 00128         BNEQ     33$                                :
                                   30         6E  D1 0012A         CMPL     CHAR, #48                          : 1391
                                              05  19 0012D         BLSS     23$                                :
                                   39         6E  D1 0012F         CMPL     CHAR, #57                          :
                                              09  15 00132         BLEQ     24$                                :
                                   63    FF   A4  9E 00134  23$:   MOVAB    -1(R4), (R3)                       : 1394
                                              52  DD 00138         PUSHL    R2                                 : 1395
                                   67         01  FB 0013A         CALLS    #1, DBG$SYNTAX_ERROR               :
                                         04   AE  7C 0013D  24$:   CLRQ     VALUE                              : 1403
                                              55  D4 00140         CLRL     TRUNC_FLAG                         : 1405
                                   30         6E  D1 00142  25$:   CMPL     CHAR, #48                          : 1406
                                              1E  19 00145         BLSS     27$                                :
                                   39         6E  D1 00147         CMPL     CHAR, #57                          :
                                              19  14 0014A         BGTR     27$                                :
                                   6E         30  C2 0014C         SUBL2    #48, CHAR                          : 1408
   04   AE        6E               0A    04   AE  7A 0014F         EMUL     VALUE, #10, CHAR, VALUE            : 1409
                                         08   AE  D5 00156         TSTL     VALUE+4                            : 1410
                                              03  13 00159         BEQL     26$                                :
                                   55         01  D0 0015B         MOVL     #1, TRUNC_FLAG                     :
                                              62  B7 0015E  26$:   DECW     (R2)                               : 1411
                                   6E         84  9A 00160         MOVZBL   (STRING_PTR)+, CHAR                : 1412
                                              DD  11 00163         BRB      25$                                : 1406
                       00000041    8F         6E  D1 00165  27$:   CMPL     CHAR, #65                          : 1419
                                              03  18 0016C  28$:   BGEQ     30$                                :
                                            00C7  31 0016E  29$:   BRW      47$                                :
                       0000005A    8F         6E  D1 00171  30$:   CMPL     CHAR, #90                          :
                                              F4  14 00178  31$:   BGTR     29$                                :
                                              5E  DD 0017A         PUSHL    SP                                 : 1421
                                              01  DD 0017C         PUSHL    #1                                 :
                                              02  DD 0017E         PUSHL    #2                                 :
                                   00028AAA   8F  DD 00180         PUSHL    #166570                            :
                                            0099  31 00186  32$:   BRW      45$                                :
                                   10         50  91 00189  33$:   CMPB     R0, #16                            : 1425
```

```
                                       03  13 0018C        BEQL    34$                           :
                                     0096  31 0018E        BRW     46$                           :
                      30               6E  D1 00191 34$:   CMPL    CHAR, #48                     : 1430
                                       05  19 00194        BLSS    35$                           :
                      39               6E  D1 00196        CMPL    CHAR, #57                     :
                                       1B  15 00199        BLEQ    37$                           :
          00000041    8F               6E  D1 0019B 35$:   CMPL    CHAR, #65                     : 1431
                                       09  19 001A2        BLSS    36$                           :
          00000046    8F               6E  D1 001A4        CMPL    CHAR, #70                     :
                                       09  15 001AB        BLEQ    37$                           :
                      63       FF      A4  9E 001AD 36$:   MOVAB   -1(R4), (R3)                  : 1434
                                       52  DD 001B1        PUSHL   R2                            : 1435
                      67               01  FB 001B3        CALLS   #1, DBG$SYNTAX_ERROR          :
                               04      AE  7C 001B6 37$:   CLRQ    VALUE                         : 1443
                                       55  D4 001B9        CLRL    TRUNC_FLAG                    : 1445
                      50               6E  D0 001BB 38$:   MOVL    CHAR, R0                      : 1446
                                       51  D4 001BE        CLRL    R1                            :
                      30               50  D1 001C0        CMPL    R0, #48                       :
                                       07  19 001C3        BLSS    39$                           :
                                       51  D6 001C5        INCL    R1                            :
                      39               50  D1 001C7        CMPL    R0, #57                       :
                                       12  15 001CA        BLEQ    40$                           :
          00000041    8F               50  D1 001CC 39$:   CMPL    R0, #65                       : 1447
                                       2F  19 001D3        BLSS    44$                           :
          00000046    8F               50  D1 001D5        CMPL    R0, #70                       :
                                       26  14 001DC        BGTR    44$                           :
                      0A               51  E9 001DE 40$:   BLBC    R1, 41$                       : 1449
                      39               50  D1 001E1        CMPL    R0, #57                       :
                                       05  14 001E4        BGTR    41$                           :
                      6E               30  C2 001E6        SUBL2   #48, CHAR                     : 1451
                                       03  11 001E9        BRB     42$                           :
                      6E               37  C2 001EB 41$:   SUBL2   #55, CHAR                     : 1453
  04    AE        6E  10       04      AE  7A 001EE 42$:   EMUL    VALUE, #16, CHAR, VALUE       : 1454
                               08      AE  D5 001F5        TSTL    VALUE+4                       : 1455
                                       03  13 001F8        BEQL    43$                           :
                      55               01  D0 001FA        MOVL    #1, TRUNC_FLAG                :
                                       62  B7 001FD 43$:   DECW    (R2)                          : 1456
                      6E               84  9A 001FF        MOVZBL  (STRING_PTR)+, CHAR           : 1457
                                       B7  11 00202        BRB     38$                           : 1446
          00000047    8F               6E  D1 00204 44$:   CMPL    CHAR, #71                     : 1463
                                       2B  19 0020B        BLSS    47$                           :
          0000005A    8F               6E  D1 0020D        CMPL    CHAR, #90                     :
                                       22  14 00214        BGTR    47$                           :
                                       5E  DD 00216        PUSHL   SP                            : 1465
                                       01  DD 00218        PUSHL   #1                            :
                                       02  DD 0021A        PUSHL   #2                            :
                          00028ABA     8F  DD 0021C        PUSHL   #166586                       :
                      66               04  FB 00222 45$:   CALLS   #4, LIB$SIGNAL                :
                                       11  11 00225        BRB     47$                           : 1303
                          00000000'    EF  9F 00227 46$:   PUSHAB  P.ABM                         : 1471
                                       01  DD 0022D        PUSHL   #1                            :
                          00028362     8F  DD 0022F        PUSHL   #164706                       :
                      66               03  FB 00235        CALLS   #3, LIB$SIGNAL                :
                                       62  B6 00238 47$:   INCW    (R2)                          : 1480
                      63       FF      A4  9E 0023A        MOVAB   -1(R4), (R3)                  : 1481
                      09               55  E9 0023E        BLBC    TRUNC_FLAG, 48$               : 1486
                          00028043     8F  DD 00241        PUSHL   #163907                       :
```

```
                66                01  FB 00247        CALLS   #1, LIB$SIGNAL                  ;
                           04     AE  D5 0024A  48$:  TSTL    VALUE                           ; 1487
                                  05  18 0024D        BGEQ    49$                             ;
           04   AE         04     AE  CE 0024F        MNEGL   VALUE, VALUE                    ;
           08   BC         04     AE  D0 00254  49$:  MOVL    VALUE, @RESULT                  ; 1488
                50                01  D0 00259        MOVL    #1, R0                          ; 1489
                                      04 0025C        RET                                     ; 1491

; Routine Size:  605 bytes,    Routine Base:  DBG$CODE + 0777
```

```
GLOBAL ROUTINE DBG$NSAVE_STRING(INPUT_DESC, BUFF_PTR, MESSAGE_VECT) =
!
! FUNCTION
!       This routine accepts the next string from the input buffer
!       and stores it away in a dynamic buffer. Leading white space is ignored.
!       Trailing white space, comma, or <cr> terminate the string.
!       The saved string is stored as a counted string.
!
! FORMAL PARAMETERS:
!
!       INPUT_DESC      - The address of a VAX standard ASCII string
!                         descriptor which describes input user command.
!
!       BUFF_PTR        - The address of a longword to contain the beginning
!                         address of the stored counted string.
!
!       MESSAGE_VECT    - The address of a longword to contain the address
!                         of a message argument vector for errors.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       On success, the stored counted ASCII string.
!
!       On failure, a message argument vector.
!
! ROUTINE VALUE:
!
!       An unsigned integer longword completion code
!
! COMPLETION CODES:
!
!       STS$K_SUCCESS (1)       - Success. String isolated and stored.
!
!       STS$K_SEVERE  (4)       - Failure. No string stored. Message argument
!                                 vector returned.
!
!                                 Note that this routine returns failure if a
!                                 string cannot be found. A NEEDMORE
!                                 message is constructed in such cases.
!
! SIDE EFFECTS:
!
!       The input buffer is updated to reflect one character beyond the last
!       character accepted.
!

    BEGIN

    MAP
        INPUT_DESC : REF DBG$STG_DESC;  ! Input string descriptor

    BIND
        DBG$CS_CR = UPLIT BYTE(1, DBG$K_CAR_RETURN);
```

```
LOCAL
     STRING: REF VECTOR[,BYTE],         ! Will hold the string
     COUNT,                             ! Count of characters in string
     POINTER;                           ! Temporary pointer


! Check for null input
!
IF DBG$NMATCH (.INPUT_DESC, DBG$CS_CR, 1)
THEN
     BEGIN
     .MESSAGE_VECT = DBG$NMAKE_ARG_VECT (DBG$_NEEDMORE);
     RETURN STS$K_SEVERE;
     END;


! Delete leading white space
!
WHILE CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) EQL ' ' DO
     BEGIN
     INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.INPUT_DESC [DSC$A_POINTER], 1);
     INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
     END;


! Save the start of the string
!
POINTER = .INPUT_DESC [DSC$A_POINTER];


! Now look for a comma, a blank, or <cr>
!
WHILE CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) NEQ ' ' AND CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) NEQ ','
     AND CH$RCHAR (.INPUT_DESC [DSC$A_POINTER]) NEQ DBG$K_CAR_RETURN DO
     BEGIN
     INPUT_DESC [DSC$A_POINTER] = CH$PLUS (.INPUT_DESC [DSC$A_POINTER], 1);
     INPUT_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH] - 1;
     END;


! Figure out how many characters to save
!
COUNT = CH$DIFF (.INPUT_DESC [DSC$A_POINTER], .POINTER);


! Allocate the proper size buffer
!
STRING = DBG$GET_TEMPMEM ((.COUNT/%UPVAL) + 1);


! Copy over the count and the characters.
!
STRING [0] = .COUNT;
CH$MOVE (.COUNT, .POINTER, STRING [1]);
```

```
; 1482        1606 2
; 1483        1607 2      ! Set up the output parameter and return
; 1484        1608 2      !
; 1485        1609 2      .BUFF_PTR = STRING [0];
; 1486        1610 2      RETURN STS$K_SUCCESS;
; 1487        1611 2
; 1488        1612 1      END;


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

                              0D  01  0012C P.ABN:  .BYTE   1, 13                                      ;

                                        DBG$CS_CR=            P.ABN


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

                               007C 00000         .ENTRY  DBG$NSAVE_STRING, Save R2,R3,R4,R5,R6      ; 1492
                                 01 DD 00002      PUSHL   #1                                         ; 1559
                   00000000'     EF 9F 00004      PUSHAB  DBG$CS_CR                                  ;
              52         04      AC D0 0000A      MOVL    INPUT_DESC, R2                             ;
                                 52 DD 0000E      PUSHL   R2                                         ;
        FB56  CF                 03 FB 00010      CALLS   #3, DBG$NMATCH                             ;
              15                 50 E9 00015      BLBC    R0, 1$                                     ;
                   000280D0      8F DD 00018      PUSHL   #164048                                    ; 1562
   00000000G  00                 01 FB 0001E      CALLS   #1, DBG$NMAKE_ARG_VECT                     ;
          0C  BC                 50 D0 00025      MOVL    R0, @MESSAGE_VECT                          ;
              50                 04 D0 00029      MOVL    #4, R0                                     ; 1563
                                    04 0002C      RET                                                ;
              50         04      A2 9E 0002D 1$:  MOVAB   4(R2), R0                                  ; 1569
              20         00      B0 91 00031 2$:  CMPB    @0(R0), #32                                ;
                                 06 12 00035      BNEQ    3$                                         ;
                                 60 D6 00037      INCL    (R0)                                       ; 1571
                                 62 B7 00039      DECW    (R2)                                       ; 1572
                                 F4 11 0003B      BRB     2$                                         ; 1569
              53                 60 D0 0003D 3$:  MOVL    (R0), POINTER                              ; 1578
              20         00      B0 91 00040 4$:  CMPB    @0(R0), #32                                ; 1583
                                 12 13 00044      BEQL    5$                                         ;
              2C         00      B0 91 00046      CMPB    @0(R0), #44                                ;
                                 0C 13 0004A      BEQL    5$                                         ;
              0D         00      B0 91 0004C      CMPB    @0(R0), #13                                ; 1584
                                 06 13 00050      BEQL    5$                                         ;
                                 60 D6 00052      INCL    (R0)                                       ; 1586
                                 62 B7 00054      DECW    (R2)                                       ; 1587
                                 E8 11 00056      BRB     4$                                         ; 1583
        52    60                 53 C3 00058 5$:  SUBL3   POINTER, (R0), COUNT                       ; 1593
        50    52                 04 C7 0005C      DIVL3   #4, COUNT, R0                              ; 1598
                         01      A0 9F 00060      PUSHAB  1(R0)                                      ;
   00000000G  00                 01 FB 00063      CALLS   #1, DBG$GET_TEMPMEM                        ;
              56                 50 D0 0006A      MOVL    R0, STRING                                 ;
              66                 52 90 0006D      MOVB    COUNT, (STRING)                            ; 1603
  01    A6    63                 52 28 00070      MOVC3   COUNT, (POINTER), 1(STRING)                ; 1604
          08  BC                 56 D0 00075      MOVL    STRING, @BUFF_PTR                          ; 1609
              50                 01 D0 00079      MOVL    #1, R0                                     ; 1610
                                    04 0007C      RET                                                ; 1612
```

; Routine Size: 125 bytes, Routine Base: DBG$CODE + 09D4

```
GLOBAL ROUTINE DBG$NGET_DIR_LIST (LEX_STG_DESC, RESULT_PTR, MESSAGE_VECT) =
!
! FUNCTION
!
!       Parse a directory list. Used by the command
!       SET SOURCE dir-list
!
! INPUTS
!
!       LEX_STG_DESC    - A string descriptor for the unparsed
!                         directory list.
!       RESULT_PTR      - The result of the parse will be placed at the
!                         address given by result_ptr.
!       MESSAGE_VECT    - Address of a longword to contain the address of
!                         the message argument vector.
!
! OUTPUTS
!
!       This routine builds a linked list of directory names and
!       leaves a pointer to this list in result_ptr.
!
! ROUTINE VALUE
!
!       A completion code which is one of either:
!       STS$K_SEVERE (4)        Unsuccessful parse
!       STS$K_SUCCESS (1)       Successful parse
!
! ALGORITHM
!
!       The directory list will be of the form:
!       [dir1]file1,[dir2]file2,...[dirn]filen
!       Do until end of list is reached:
!           Pick up a single directory name by scanning until
!           a separating comma is reached (commas inside of
!           square brackets are ignored)
!           Allocate space for a node containing the directory name.
!           Copy the directory name into this node, and
!           link the node into the list.
!

    BEGIN

    MAP
        LEX_STG_DESC : REF BLOCK [, BYTE]; ! Input string descriptor

    LOCAL
        TOT_LENGTH,                     ! Number of chars to be read from
                                        !       PARSE_STG_DESC
        TOT_CHARS_READ,                 ! Number of chars read from
                                        !       PARSE_STG_DESC so far
        CHARS_READ,                     ! Number of chars in directory name
                                        !       under construction
        IN_BRACKETS,                    ! TRUE if we are inside of square
                                        !       brackets
        CHAR,                           ! Current character
        PREV_CHAR_PTR,                  ! ???
        FIRST_DIR_FLAG,                 ! TRUE if this is first directory in
```

```
                                                      !      list
        PREV_DIRNAME: REF SDSL$ENTRY,                 ! Previous directory name
        CURR_DIRNAME: REF SDSL$ENTRY,                 ! Current directory name
        FIRST_DIRNAME,                                ! Pointer to first directory name
                                                      !      (head of linked list)
        INPUT_PTR,                                    ! ???
        OUTPUT_PTR,                                   ! ???
        NEW_INPUT_PTR,                                ! ???
        PREV_CHAR,                                    ! ???
        QUOTE_FLAG,                                   ! TRUE if the directory currently being
                                                      !      processed is inside of quotes
        DBLQUOTE_FLAG;                                ! TRUE if the directory currently being
                                                      !      processed is inside double quotes



    ! Initialize the various flags and counters we need.
    !
    IN_BRACKETS = FALSE;
    TOT_LENGTH = .LEX_STG_DESC [DSC$W_LENGTH];
    TOT_CHARS_READ = 0;
    QUOTE_FLAG = FALSE;
    DBLQUOTE_FLAG = FALSE;
    CHARS_READ = 0;


    ! FIRST_DIR_FLAG is true initially because we are processing the
    ! first directory in the list.
    !
    FIRST_DIR_FLAG = TRUE;
    PREV_DIRNAME = 0;
    WHILE .TOT_CHARS_READ LSS .TOT_LENGTH DO
        BEGIN
        INPUT_PTR = CH$PTR (.LEX_STG_DESC [DSC$A_POINTER]);
        CHAR = CH$RCHAR_A (INPUT_PTR);

        !
        ! Skip leading blanks
        !
        WHILE .CHAR EQL DBG$K_BLANK AND .TOT_CHARS_READ LSS .TOT_LENGTH DO
            BEGIN
            CHAR = CH$RCHAR_A (INPUT_PTR);
            TOT_CHARS_READ = .TOT_CHARS_READ + 1;
            LEX_STG_DESC [DSC$A_POINTER] =
                                   CH$PLUS(.LEX_STG_DESC [DSC$A_POINTER], 1);
            LEX_STG_DESC [DSC$W_LENGTH] = .LEX_STG_DESC [DSC$W_LENGTH] - 1;
            END;

        IF .CHAR EQL DBG$K_QUOTE OR .CHAR EQL DBG$K_DBLQUOTE
        THEN
            BEGIN
            IF .CHAR EQL DBG$K_QUOTE
            THEN
                QUOTE_FLAG = TRUE

            ELSE
                DBLQUOTE_FLAG = TRUE;
```

```
4
4
4              ! Advance the pointer past the quote.
4              !
4              LEX_STG_DESC [DSC$A_POINTER] =
4                                   CH$PLUS(.LEX_STG_DESC [DSC$A_POINTER], 1);
4              LEX_STG_DESC [DSC$W_LENGTH] = .LEX_STG_DESC [DSC$W_LENGTH] - 1;
4              TOT_CHARS_READ = .TOT_CHARS_READ + 1;
4              CHAR = CH$RCHAR_A (INPUT_PTR);
3              END;
3
3
3          ! Pick up next directory name.
3          !
3          WHILE .TOT_CHARS_READ LSS .TOT_LENGTH AND
3                .CHAR NEQ DBG$K_CAR_RETURN AND
4                (.CHAR NEQU %C',' OR .IN_BRACKETS)
3          DO
4                  BEGIN
4              IF .CHAR EQLU %C'[' OR .CHAR EQLU %C'<' THEN IN_BRACKETS = TRUE;
4              IF .CHAR EQLU %C']' OR .CHAR EQLU %C'>' THEN IN_BRACKETS = FALSE;
4              IF .CHAR NEQ DBG$K_BLANK
4              THEN
4                  PREV_CHAR_PTR = CH$PLUS (.INPUT_PTR, -1);
4
4              PREV_CHAR = .CHAR;
4              CHAR = CH$RCHAR_A (INPUT_PTR);
4
4
4              ! If this is a doubled up quote character, undouble them and mark
4              ! the position.
4              !
4              IF .DBLQUOTE_FLAG           AND
4                 .CHAR EQL DBG$K_DBLQUOTE AND
4                 .PREV_CHAR EQL DBG$K_DBLQUOTE
4              THEN
4                  (.PREV_CHAR_PTR)<0, 8, 0> = 0;
4
4              TOT_CHARS_READ = .TOT_CHARS_READ + 1;
4              CHARS_READ = .CHARS_READ + 1;
3              END;                        ! End of inner while loop
3
3
3          ! We now have a complete directory name.  Allocate space for it.
3          !
3          CURR_DIRNAME = DBG$GET_MEMORY (2 + (1 + .CHARS_READ)/%UPVAL);
4          IF (.FIRST_DIR_FLAG)
3          THEN
4              BEGIN
4              FIRST_DIR_FLAG = FALSE;
4              FIRST_DIRNAME = .CURR_DIRNAME;
3              END;
3
3
3          ! Link in next directory name
3          !
4          IF (.PREV_DIRNAME NEQA 0)
```

```
        THEN
            PREV_DIRNAME [SDSL$L_ENT_FLINK] = .CURR_DIRNAME;


        ! Set PREV_DIRNAME for next time around loop.
        !
        PREV_DIRNAME = .CURR_DIRNAME;


        ! Fill in the fields of CURR_DIRNAME.
        !
        CURR_DIRNAME [SDSL$L_ENT_FLINK] = 0;
        CURR_DIRNAME [SDSL$B_ENT_DIRLEN] = 0;
        NEW_INPUT_PTR = CH$PTR (.LEX_STG_DESC [DSC$A_POINTER]);
        OUTPUT_PTR = CH$PTR (.CURR_DIRNAME + 5);
        INCR I FROM 0 TO .CHARS_READ - 1 DO
            BEGIN
            CHAR = CH$RCHAR_A (NEW_INPUT_PTR);
            IF .CHAR NEQ 0
            THEN
                BEGIN
                CH$WCHAR_A (.CHAR, OUTPUT_PTR);
                CURR_DIRNAME [SDSL$B_ENT_DIRLEN] =
                                .CURR_DIRNAME [SDSL$B_ENT_DIRLEN] + 1;
                END;

            END;


        ! Strip off trailing quote.
        !
        IF .QUOTE_FLAG
        THEN
            BEGIN
            IF CH$RCHAR (.PREV_CHAR_PTR) EQL DBG$K_QUOTE
            THEN
                BEGIN
                CURR_DIRNAME [SDSL$B_ENT_DIRLEN] = .CURR_DIRNAME[SDSL$B_ENT_DIRLEN]
                                    - (.INPUT_PTR - .PREV_CHAR_PTR - 1);
                QUOTE_FLAG = FALSE;
                END

            END


        ! ???
        !
        ELSE IF .DBLQUOTE_FLAG
        THEN
            BEGIN
            IF CH$RCHAR (.PREV_CHAR_PTR) EQL DBG$K_DBLQUOTE
            THEN
                BEGIN
                CURR_DIRNAME [SDSL$B_ENT_DIRLEN] = .CURR_DIRNAME [SDSL$B_ENT_DIRLEN] - (.INPUT_PTR -
                .PREV_CHAR_PTR - 1);
                DBLQUOTE_FLAG = FALSE;
                END;
```

```
; 1718          1841  4
; 1719          1842  3              END;
; 1720          1843  3
; 1721          1844  3
; 1722          1845  3         ! If there are still characters left in the input string, increment
; 1723          1846  3         ! CHARS_READ to take us past the separating comma.
; 1724          1847  3         !
; 1725          1848  3         IF .TOT_CHARS_READ LSS .TOT_LENGTH
; 1726          1849  3         THEN
; 1727          1850  4              BEGIN
; 1728          1851  4              CHARS_READ = .CHARS_READ + 1;
; 1729          1852  4              TOT_CHARS_READ = .TOT_CHARS_READ + 1;
; 1730          1853  3              END;
; 1731          1854  3
; 1732          1855  3         LEX_STG_DESC [DSC$W_LENGTH] = .LEX_STG_DESC [DSC$W_LENGTH] - .CHARS_READ;
; 1733          1856  3         LEX_STG_DESC [DSC$A_POINTER] = CH$PLUS (.LEX_STG_DESC [DSC$A_POINTER], .CHARS_READ);
; 1734          1857  3         CHARS_READ = 0;
; 1735          1858  2         END;                           ! End of outer while loop
; 1736          1859  2
; 1737          1860  2    IF .QUOTE_FLAG OR .DBLQUOTE_FLAG THEN SIGNAL (DBG$_MATQUOMIS);
; 1738          1861  2
; 1739          1862  2
; 1740          1863  2    ! Place pointer to linked list into into the result pointer.
; 1741          1864  2    .RESULT_PTR = .FIRST_DIRNAME;
; 1742          1865  2    RETURN STS$K_SUCCESS
; 1743          1866  2
; 1744          1867  1    END;



                         0FFC 00000          .ENTRY   DBG$NGET_DIR_LIST, Save R2,R3,R4,R5,R6,R7,-  ; 1613
                                                      R8,R9,R10,R11                                 ;
            5E        20   C2 00002          SUBL2    #32, SP                                       ;
            52   04   AC   D0 00005          MOVL     LEX_STG_DESC, R2                              ; 1689
            5A        62   3C 00009          MOVZWL   (R2), TOT_LENGTH                              ;
                      55   D4 0000C          CLRL     TOT_CHARS_READ                                ; 1690
                      59   D4 0000E          CLRL     QUOTE_FLAG                                    ; 1691
                      5B   D4 00010          CLRL     DBLQUOTE_FLAG                                 ; 1692
                      57   D4 00012          CLRL     CHARS_READ                                    ; 1693
       18   AE        01   7D 00014          MOVQ     #1, FIRST_DIR_FLAG                            ; 1699
                 14   AE   D4 00018          CLRL     PREV_DIRNAME                                  ; 1700
            53   04   A2   9E 0001B          MOVAB    4(R2), R3                                     ; 1703
            5A        55   D1 0001F 1$:      CMPL     TOT_CHARS_READ, TOT_LENGTH                    ; 1701
                      03   19 00022          BLSS     2$                                            ;
                    013C   31 00024          BRW      25$                                           ;
            56        63   D0 00027 2$:      MOVL     (R3), INPUT_PTR                               ; 1703
            54        86   9A 0002A          MOVZBL   (INPUT_PTR)+, CHAR                            ; 1704
            20        54   D1 0002D 3$:      CMPL     CHAR, #32                                     ; 1709
                      10   12 00030          BNEQ     4$                                            ;
            5A        55   D1 00032          CMPL     TOT_CHARS_READ, TOT_LENGTH                    ;
                      0B   18 00035          BGEQ     4$                                            ;
            54        86   9A 00037          MOVZBL   (INPUT_PTR)+, CHAR                            ; 1711
                      55   D6 0003A          INCL     TOT_CHARS_READ                                ; 1712
                      63   D6 0003C          INCL     (R3)                                          ; 1714
                      62   B7 0003E          DECW     (R2)                                          ; 1715
```

```
                                   EB  11 00040         BRB     3$                                       ; 1709
                                   51  D4 00042 4$:     CLRL    R1                                       ; 1718
                     27            54  D1 00044         CMPL    CHAR, #39                                ;
                                   04  12 00047         BNEQ    5$                                       ;
                                   51  D6 00049         INCL    R1                                       ;
                                   05  11 0004B         BRB     6$                                       ;
                     22            54  D1 0004D 5$:     CMPL    CHAR, #34                                ;
                                   14  12 00050         BNEQ    9$                                       ;
                     05            51  E9 00052 6$:     BLBC    R1, 7$                                   ; 1721
                     59            01  D0 00055         MOVL    #1, QUOTE_FLAG                           ; 1723
                                   03  11 00058         BRB     8$                                       ;
                     5B            01  D0 0005A 7$:     MOVL    #1, DBLQUOTE_FLAG                        ; 1726
                                   63  D6 0005D 8$:     INCL    (R3)                                     ; 1732
                                   62  B7 0005F         DECW    (R2)                                     ; 1733
                                   55  D6 00061         INCL    TOT_CHARS_READ                           ; 1734
                     54            86  9A 00063         MOVZBL  (INPUT_PTR)+, CHAR                       ; 1735
                     5A            55  D1 00066 9$:     CMPL    TOT_CHARS_READ, TOT_LENGTH               ; 1741
                                   57  18 00069         BGEQ    17$                                      ;
                     0D            54  D1 0006B         CMPL    CHAR, #13                                ; 1742
                                   52  13 0006E         BEQL    17$                                      ;
                     2C            54  D1 00070         CMPL    CHAR, #44                                ; 1743
                                   04  12 00073         BNEQ    10$                                      ;
                     49      1C    AE  E9 00075         BLBC    IN_BRACKETS, 17$                         ;
          0000005B   8F            54  D1 00079 10$:    CMPL    CHAR, #91                                ; 1746
                                   05  13 00080         BEQL    11$                                      ;
                     3C            54  D1 00082         CMPL    CHAR, #60                                ;
                                   04  12 00085         BNEQ    12$                                      ;
                1C   AE            01  D0 00087 11$:    MOVL    #1, IN_BRACKETS                          ;
          0000005D   8F            54  D1 0008B 12$:    CMPL    CHAR, #93                                ; 1747
                                   05  13 00092         BEQL    13$                                      ;
                     3E            54  D1 00094         CMPL    CHAR, #62                                ;
                                   03  12 00097         BNEQ    14$                                      ;
                             1C    AE  D4 00099 13$:    CLRL    IN_BRACKETS                              ;
                     20            54  D1 0009C 14$:    CMPL    CHAR, #32                                ; 1748
                                   04  13 0009F         BEQL    15$                                      ;
                     58      FF    A6  9E 000A1         MOVAB   -1(R6), PREV_CHAR_PTR                    ; 1750
                0C   AE            54  D0 000A5 15$:    MOVL    CHAR, PREV_CHAR                          ; 1752
                     54            86  9A 000A9         MOVZBL  (INPUT_PTR)+, CHAR                       ; 1753
                     0D            5B  E9 000AC         BLBC    DBLQUOTE_FLAG, 16$                       ; 1759
                     22            54  D1 000AF         CMPL    CHAR, #34                                ; 1760
                                   08  12 000B2         BNEQ    16$                                      ;
                     22      0C    AE  D1 000B4         CMPL    PREV_CHAR, #34                           ; 1761
                                   02  12 000B8         BNEQ    16$                                      ;
                                   68  94 000BA         CLRB    (PREV_CHAR_PTR)                          ; 1763
                                   55  D6 000BC 16$:    INCL    TOT_CHARS_READ                           ; 1765
                                   57  D6 000BE         INCL    CHARS_READ                               ; 1766
                                   A4  11 000C0         BRB     9$                                       ; 1741
                     51      01    A7  9E 000C2 17$:    MOVAB   1(R7), R1                                ; 1772
                     51            04  C6 000C6         DIVL2   #4, R1                                   ;
                             02    A1  9F 000C9         PUSHAB  2(R1)                                    ;
          00000000G  00            01  FB 000CC         CALLS   #1, DBG$GET_MEMORY                       ;
                     07      18    AE  E9 000D3         BLBC    FIRST_DIR_FLAG, 18$                      ; 1773
                             18    AE  D4 000D7         CLRL    FIRST_DIR_FLAG                           ; 1776
                04   AE            50  D0 000DA         MOVL    CURR_DIRNAME, FIRST_DIRNAME              ; 1777
                             14    AE  D5 000DE 18$:    TSTL    PREV_DIRNAME                             ; 1783
                                   04  13 000E1         BEQL    19$                                      ;
                14   BE            50  D0 000E3         MOVL    CURR_DIRNAME, @PREV_DIRNAME              ; 1785
```

```
                 14   AE              50  D0 000E7 19$:   MOVL    CURR_DIRNAME, PREV_DIRNAME          ; 1790
                                      60  D4 000EB        CLRL    (CURR_DIRNAME)                      ; 1795
                            04        A0  94 000ED        CLRB    4(CURR_DIRNAME)                     ; 1796
                 08   AE              63  D0 000F0        MOVL    (R3), NEW_INPUT_PTR                 ; 1797
                 10   AE    05        A0  9E 000F4        MOVAB   5(R0), OUTPUT_PTR                   ; 1798
                      51              01  CE 000F9        MNEGL   #1, I                               ; 1799
                      15              11     000FC        BRB     21$                                 ;
                      54    08        BE  9A 000FE 20$:   MOVZBL  @NEW_INPUT_PTR, CHAR                ; 1801
                            08        AE  D6 00102        INCL    NEW_INPUT_PTR                       ;
                                      54  D5 00105        TSTL    CHAR                                ; 1802
                                      0A  13 00107        BEQL    21$                                 ;
                 10   BE              54  90 00109        MOVB    CHAR, @OUTPUT_PTR                   ; 1805
                            10        AE  D6 0010D        INCL    OUTPUT_PTR                          ;
                            04        A0  96 00110        INCB    4(CURR_DIRNAME)                     ; 1807
            E7        51              57  F2 00113 21$:   AOBLSS  CHARS_READ, I, 20$                  ; 1799
                      1A              59  E9 00117        BLBC    QUOTE_FLAG, 22$                     ; 1815
                      27              68  91 0011A        CMPB    (PREV_CHAR_PTR), #39                ; 1818
                      30              12     0011D        BNEQ    23$                                 ;
            51        56              58  C3 0011F        SUBL3   PREV_CHAR_PTR, INPUT_PTR, R1        ; 1822
                      6E    04        A0  9A 00123        MOVZBL  4(CURR_DIRNAME), (SP)               ;
            51        6E              51  C3 00127        SUBL3   R1, (SP), R1                        ;
       04   A0        51              01  81 0012B        ADDB3   #1, R1, 4(CURR_DIRNAME)             ;
                                      59  D4 00130        CLRL    QUOTE_FLAG                          ; 1823
                                      1B  11 00132        BRB     23$                                 ; 1817
                      18              5B  E9 00134 22$:   BLBC    DBLQUOTE_FLAG, 23$                  ; 1831
                      22              68  91 00137        CMPB    (PREV_CHAR_PTR), #34                ; 1834
                      13              12     0013A        BNEQ    23$                                 ;
            51        56              58  C3 0013C        SUBL3   PREV_CHAR_PTR, INPUT_PTR, R1        ; 1838
                      6E    04        A0  9A 00140        MOVZBL  4(CURR_DIRNAME), (SP)               ; 1837
            51        6E              51  C3 00144        SUBL3   R1, (SP), R1                        ;
       04   A0        51              01  81 00148        ADDB3   #1, R1, 4(CURR_DIRNAME)             ;
                                      5B  D4 0014D        CLRL    DBLQUOTE_FLAG                       ; 1839
                      5A              55  D1 0014F 23$:   CMPL    TOT_CHARS_READ, TOT_LENGTH          ; 1848
                                      04  18 00152        BGEQ    24$                                 ;
                                      57  D6 00154        INCL    CHARS_READ                          ; 1851
                                      55  D6 00156        INCL    TOT_CHARS_READ                      ; 1852
                      62              57  A2 00158 24$:   SUBW2   CHARS_READ, (R2)                    ; 1855
                      63              57  C0 0015B        ADDL2   CHARS_READ, (R3)                    ; 1856
                                      57  D4 0015E        CLRL    CHARS_READ                          ; 1857
                                    FEBC  31 00160        BRW     1$                                  ; 1701
                      03              59  E8 00163 25$:   BLBS    QUOTE_FLAG, 26$                     ; 1860
                      0D              5B  E9 00166        BLBC    DBLQUOTE_FLAG, 27$                  ;
                          00028E30    8F  DD 00169 26$:   PUSHL   #167472                             ;
          00000000G   00              01  FB 0016F        CALLS   #1, LIB$SIGNAL                      ;
                 08   BC    04        AE  D0 00176 27$:   MOVL    FIRST_DIRNAME, @RESULT_PTR          ; 1864
                      50              01  D0 0017B        MOVL    #1, R0                              ; 1865
                                          04 0017E        RET                                         ; 1867

; Routine Size:  383 bytes,    Routine Base:  DBG$CODE + 0A51
```

```
GLOBAL ROUTINE DBG$EXPAND_DEFINE_NAME(INPUT_DESC, KIND, RESULT_ADDR) =
!
! FUNCTION
!       This routine checks for the next symbol in the input stream being
!       a name which has been define with the DEFINE command. If so, and
!       if the kind matches the kind given in the input parameter, then
!       the corresponding value is returned in result_addr.
!
! INPUTS
!       INPUT_DESC -    A string descriptor for the remaining input.
!
!       KIND -          The kind of DEFINE symbol we are expecting (one
!                       of define_address, define_command, define_procedure,
!                       define_string, or define_value)
!
!       RESULT_ADDR -   The address in which to leave the result
!
! OUTPUTS
!       The routine value is one of:
!       TRUE -  A matching symbol was found
!       FALSE - A matching symbol was not found
!
!       If TRUE, the output parameter RESULT_ADDR is filled in.
!       Also, the input descriptor is updated to point past the name
!       that was read.
!

    BEGIN

    MAP
        INPUT_DESC: REF BLOCK[,BYTE];   ! Input string descriptor

    LOCAL
        FOUND_FLAG,                     ! TRUE if we found a DEFINEd
                                        !       symbol matching the input
        GLOBAL_FLAG,                    ! Holds an output value for
                                        !       DBG$DEF_SYM_FIND
        MESSAGE_VECT,                   ! Dummy error message vector
        NAME_PTR: REF VECTOR[,BYTE],    ! Points to a name which
                                        !       could potentially be
                                        !       a DEFINEd symbol.
        SAVED_INPUT_LENGTH,             ! Copy of the length from the
                                        !       original input descriptor
        SAVED_INPUT_POINTER,            ! Copy of the pointer from the
                                        !       original input descriptor
        SYMBOL_KIND,                    ! Kind of defined symbol
        SYMBOL_VALUE;                   ! Value of DEFINEd symbol



    ! Initialize found_flag.
    !
    FOUND_FLAG = FALSE;


    ! First save away the input descriptor.
    !
```

```
SAVED_INPUT_POINTER = .INPUT_DESC [DSC$A_POINTER];
SAVED_INPUT_LENGTH = .INPUT_DESC [DSC$W_LENGTH];


! Now attempt to read a name which potentially could be DEFINEd.
!
IF DBG$NREAD_NAME (.INPUT_DESC, NAME_PTR, MESSAGE_VECT)
THEN
    BEGIN


    ! If we read a name, we now attempt to look it up.
    !
    IF DBG$DEF_SYM_FIND (.NAME_PTR, SYMBOL_KIND,
                         SYMBOL_VALUE, GLOBAL_FLAG, MESSAGE_VECT)
    THEN
        BEGIN


        ! Found a definition. Check the type of symbol. We expect
        ! the kind to match what is given in "kind" at this point.
        !
        IF .SYMBOL_KIND EQL .KIND
        THEN
            BEGIN
            .RESULT_ADDR = .SYMBOL_VALUE;
            FOUND_FLAG = TRUE;
            END;

        END;


    ! Deallocate the space for the name.
    !
    DBG$REL_MEMORY(.NAME_PTR);
    END;


! If we found a symbol, return success.  Otherwise, back up the input
! descriptor so that it will be returned unchanged.
!
IF .FOUND_FLAG THEN RETURN TRUE;
INPUT_DESC [DSC$A_POINTER] = .SAVED_INPUT_POINTER;
INPUT_DESC [DSC$W_LENGTH] = .SAVED_INPUT_LENGTH;
RETURN FALSE;

END;
```

```
                    003C 00000        .ENTRY  DBG$EXPAND_DEFINE_NAME, Save R2,R3,R4,R5   ; 1868
  5E             14   C2 00002        SUBL2   #20, SP                                    ;
                 53   D4 00005        CLRL    FOUND_FLAG                                 ; 1920
  52       04    AC   D0 00007        MOVL    INPUT_DESC, R2                             ; 1925
  54       04    A2   D0 0000B        MOVL    4(R2), SAVED_INPUT_POINTER                 ;
```

```
            55              62 3C 0000F        MOVZWL  (R2), SAVED_INPUT_LENGTH          ; 1926
                       04   AE 9F 00012        PUSHAB  MESSAGE_VECT                      ; 1931
                       04   AE 9F 00015        PUSHAB  NAME_PTR                          ;
                            52 DD 00018        PUSHL   R2                                ;
   00000000G   00           03 FB 0001A        CALLS   #3, DBG$NREAD_NAME                ;
               31           50 E9 00021        BLBC    R0, 2$                            ;
                       04   AE 9F 00024        PUSHAB  MESSAGE_VECT                      ; 1938
                       0C   AE 9F 00027        PUSHAB  GLOBAL_FLAG                       ;
                       14   AE 9F 0002A        PUSHAB  SYMBOL_VALUE                      ;
                       1C   AE 9F 0002D        PUSHAB  SYMBOL_KIND                       ;
                       10   AE DD 00030        PUSHL   NAME_PTR                          ;
   00000000G   00           05 FB 00033        CALLS   #5, DBG$DEF_SYM_FIND              ;
               0F           50 E9 0003A        BLBC    R0, 1$                            ;
         08    AC      10   AE D1 0003D        CMPL    SYMBOL_KIND, KIND                 ; 1947
                            08 12 00042        BNEQ    1$                                ;
         0C    BC      0C   AE D0 00044        MOVL    SYMBOL_VALUE, @RESULT_ADDR        ; 1950
               53           01 D0 00049        MOVL    #1, FOUND_FLAG                    ; 1951
                            6E DD 0004C 1$:    PUSHL   NAME_PTR                          ; 1959
   00000000G   00           01 FB 0004E        CALLS   #1, DBG$REL_MEMORY                ;
               04           53 E9 00055 2$:    BLBC    FOUND_FLAG, 3$                    ; 1966
               50           01 D0 00058        MOVL    #1, R0                            ;
                               04 0005B        RET                                       ;
         04    A2           54 D0 0005C 3$:    MOVL    SAVED_INPUT_POINTER, 4(R2)        ; 1967
               62           55 B0 00060        MOVW    SAVED_INPUT_LENGTH, (R2)          ; 1968
                            50 D4 00063        CLRL    R0                                ; 1969
                               04 00065        RET                                       ; 1971

; Routine Size:  102 bytes,    Routine Base:  DBG$CODE + 0BD0
```

```
1 GLOBAL ROUTINE DBG$SYNTAX_ERROR(INPUT_DESC): NOVALUE =
1 !
1 ! FUNCTION
1 !       This routine is called when a syntax error has occurred because
1 !       an expected piece of input was not found.  The routine accepts
1 !       an input string descriptor as input which is expected to point
1 !       to the start of the offending piece of command input.  It then
1 !       signals the "need more input" message if the next character is
1 !       a carriage-return (unexpected end of input) or the "syntax error"
1 !       error message otherwise.
1 !
1 ! INPUTS
1 !       INPUT_DESC - A pointer to a string descriptor which points to the
1 !                    current parse position in the input command line.
1 !
1 ! OUTPUTS
1 !       INPUT_DESC - The input string descriptor may be updated to point
1 !                    to the first character after the offending syntactic
1 !                    entity.
1 !
1 !       Since this routine always exits by signalling out, no value is
1 !                    ever returned.  Control is not returned either.
1 !
1
2   BEGIN
2
2   BIND
2       DBG$CS_CR        = UPLIT BYTE(1, 13);      ! Carriage return character
2
2   LOCAL
2       STRDESC: BLOCK[8,BYTE],           ! String descriptor
2       STRPTR: REF VECTOR[,BYTE];        ! Pointer to error string
2
2
2
2   ! If the line has already ended, signal the "need more" error message.
2   ! Otherwise, signal the "syntax error" error message.
2   !
2   IF DBG$NMATCH(.INPUT_DESC, DBG$CS_CR, 1) THEN SIGNAL(DBG$_NEEDMORE);
2   STRPTR = DBG$NNEXT_WORD(.INPUT_DESC);
2   STRDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
2   STRDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
2   STRDESC[DSC$W_LENGTH] = .STRPTR[0];
2   STRDESC[DSC$A_POINTER] = STRPTR[1];
2   SIGNAL(DBG$_SYNTAX, 1, STRDESC);
2   RETURN;
2
1   END;


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

                OD  01  0012E P.ABO:    .BYTE   1, 13                              ;

                              DBG$CS_CR=          P.ABO
```

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

                                   0004 00000          .ENTRY  DBG$SYNTAX_ERROR, Save R2                          ; 1972
              52 00000000G    00     9E 00002          MOVAB   LIB$SIGNAL, R2                                     ;
              5E              08     C2 00009          SUBL2   #8, SP                                             ;
                              01     DD 0000C          PUSHL   #1                                                 ; 2010
                 00000000'    EF     9F 0000E          PUSHAB  DBG$CS_CR                                          ;
                       04     AC     DD 00014          PUSHL   INPUT_DESC                                         ;
        F8ED  CF              03     FB 00017          CALLS   #3, DBG$NMATCH                                     ;
              09              50     E9 0001C          BLBC    R0, 1$                                             ;
                 000280D0     8F     DD 0001F          PUSHL   #164048                                            ;
              62              01     FB 00025          CALLS   #1, LIB$SIGNAL                                     ;
                       04     AC     DD 00028 1$:      PUSHL   INPUT_DESC                                         ; 2011
        F935  CF              01     FB 0002B          CALLS   #1, DBG$NNEXT_WORD                                 ;
          02  AE         010E 8F     B0 00030          MOVW    #270, STRDESC+2                                    ; 2013
              6E              60     9B 00036          MOVZBW  (STRPTR), STRDESC                                  ; 2014
          04  AE           01 A0     9E 00039          MOVAB   1(R0), STRDESC+4                                   ; 2015
                              5E     DD 0003E          PUSHL   SP                                                 ; 2016
                              01     DD 00040          PUSHL   #1                                                 ;
                 00028238     8F     DD 00042          PUSHL   #164408                                            ;
              62              03     FB 00048          CALLS   #3, LIB$SIGNAL                                     ;
                                     04 0004B          RET                                                        ; 2019

; Routine Size:  76 bytes,    Routine Base:  DBG$CODE + 0C36


; 1899          2020  1
; 1900          2021  0 END ELUDOM



                                                       .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                          Bytes                    Attributes
;
;   DBG$OWN                              81  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   DBG$PLIT                            304  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;   DBG$CODE                           3202  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         8         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        63         4          97        00:01.9
;   _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                                418         0         0          31        00:00.3
;   _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1          386        11         2          22        00:00.3
```

```
;                               COMMAND QUALIFIERS

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

; Size:          3202 code + 385 data bytes
; Run Time:         01:03.1
; Elapsed Time:     03:14.5
; Lines/CPU Min:      1922
; Lexemes/CPU-Min: 12329
; Memory Used:  462 pages
; Compilation Complete
```

DBGNERMSG
LIS

DBGNHELP
LIS

DBGNPARSE
LIS

DBGNEXCTE
LIS

DBGNPNP
LIS