```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```

FILEID**DBGGEN

```
DDDDDDDD    BBBBBBBB      GGGGGGGG    GGGGGGGG  EEEEEEEEEE  NN       NN
DDDDDDDD    BBBBBBBB      GGGGGGGG    GGGGGGGG  EEEEEEEEEE  NN       NN
DD      DD  BB      BB  GG          GG          EE          NN       NN
DD      DD  BB      BB  GG          GG          EE          NN       NN
DD      DD  BB      BB  GG          GG          EE          NNNN     NN
DD      DD  BB      BB  GG          GG          EE          NNNN     NN
DD      DD  BBBBBBBB    GG          GG          EEEEEEEE    NN  NN   NN
DD      DD  BBBBBBBB    GG          GG          EEEEEEEE    NN  NN   NN
DD      DD  BB      BB  GG  GGGGGG  GG  GGGGGG  EE          NN    NNNN
DD      DD  BB      BB  GG  GGGGGG  GG  GGGGGG  EE          NN    NNNN
DD      DD  BB      BB  GG      GG  GG      GG  EE          NN      NN      ....
DD      DD  BB      BB  GG      GG  GG      GG  EE          NN      NN      ....
DDDDDDDD    BBBBBBBB      GGGGGG      GGGGGG    EEEEEEEEEE  NN      NN      ....
DDDDDDDD    BBBBBBBB      GGGGGG      GGGGGG    EEEEEEEEEE  NN      NN      ....

RRRRRRRR    EEEEEEEEEE    QQQQQQ
RRRRRRRR    EEEEEEEEEE    QQQQQQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RR  RR      EE          QQ  QQ  QQ
RR  RR      EE          QQ  QQ  QQ
RR    RR    EE          QQ    QQ
RR    RR    EE          QQ    QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
```

```
!       DBGGEN.REQ - Require file for VAX/VMS DEBUG facility.
!
! Version:      'V04-000'
!
!****************************************************************************
!*                                                                          *
!*  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.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!++
!       Modified by:
!               Bruce Olsen, 30 Aug 1979
!               Richard Title, 20 Aug 1981
!
!       29-Jun-80       DLP     Added literals for PL/I, PASCAL, and C
!       29-JAN-81       DLP     chs_per_lexeme is now 50
!       20-AUG-81        RT     Changed step_lvl_size and added step_source
!                               because of new command
!                               SET STEP SOURCE
!       20-JAN-82        RT     Moved CIS declarations from here to DBGLIB.
!       06-MAY-82        RT     Added structures for DEFINE
!--
```

```
MACRO
        NULL_POS_SIZE   =0, 0, 0%,      ! Null PSE for undotted references to
                                        !     blocks
        OPERAND_MODE    =0, 4, 4, 0%,   ! Mode part of an operand
        OPERAND_VALUE   =0, 0, 4, 0%;   ! Value part of an operand

LITERAL
        DBG_FAC_CODE    = 2 * 65536,    ! DEBUG facility code
        THREAD_CODE     =1,             ! Conditional switch for threaded code
        FATAL_BIT       =4,             ! Mask for fatal bit in error codes
        ADD_THE_OFFSET  =1,             ! Add offset to value
        SUB_THE_OFFSET  =0,             ! Subtract offset from value

        ! The next three literals used to be defined in SYSLIT.REC
        !
        NO_OVERRIDE     = 0,            ! Use current length and radix
        TTY_OUT_WIDTH   = 132,          ! Width of terminal line
        TTY_ASCII_LEN   = 80,           ! Max size of character variable

        ! Values for register name tables
        !
        REGISTER_COUNT  =17,            ! Seventeen registers counting PSL

        ! size parameters
        !
        MAX_ASCII_LENG  = 4096,         ! Largest N in "ASCII:N"
        MAX_STACK_PTR   =20,            ! Depth of the parser's parse stack.
        NO_OF_INP_CHARS =132,           ! Max number of characters in input line
                                        ! ***Must be divisible by 4***
        CHS_PER_LEXEME  =50,            ! Max number of characters in a single lexeme
                                        ! ***Must be divisible by 4***
        NO_OF_TOKENS    =30,            ! Max number of tokens permitted
                                        ! ***Must be an even number***
        MAX_BYTE_INT    = 127,          ! Largest byte integer
        ALL_BPTS        =-1,            ! All breakpoints in chain
        BPT_INSTRUCTION =%X'03',        ! Opcode assignment for bpt instruction (srm revision 3)
        BREAK_POINT     = 1,            ! Table entry is breakpoint
        TRACE_POINT     = 2,            ! Table entry is tracepoint
        WATCH_POINT     = 3,            ! Table entry is watchpoint

        WRITE_WATCH     = 1,            ! Watchpoint is for write access
        READ_WRIT_WATCH = 2,            ! Watchpoint is for read/write access
        EXECUTE_WATCH   = 3,            ! This is break or trace point

        NUM_MAX_LENGTH  =11,            ! Maximum number of characters per numeric string
        SYM_MAX_LENGTH  =31,            ! Maximum number of characters per symbol
        UPPER_CASE_DIF  ='A' - 'A',     ! Difference between ASCII representation of upper and lower case
        ASCII_OFFSET    =%O'60',        ! Offset from numeric value to ASCII value

        ! ASCII character representations
        !
        LINEFEED        =%O'12',        ! ASCII representation of linefeed
```

```
CARRIAGE_RET    =%O'15',        ! ASCII representation of carriage return
ASC_AT_SIGN     =%ASCII '@',    ! ASCII representation of an at sign
ASC_CLOS_PAREN  =%ASCII ')',    ! ASCII representation of closed parenthesis
ASC_COMMA       =%ASCII ',',    ! ASCII representation of a comma
ASC_DOUB_QUOTE  =%ASCII '"',    ! ASCII representation of a double quote
ASC_MINUS       =%ASCII '-',    ! ASCII representation of a minus sign
ASC_OPEN_PAREN  =%ASCII '(',    ! ASCII representation of open parenthesis
ASC_PERCENT     =%ASCII '%',    ! ASCII representation of a percent sign
ASC_PERIOD      =%ASCII '.',    ! ASCII representation of a period
ASC_PLUS        =%ASCII '+',    ! ASCII representation of a plus sign
ASC_POUNDS      =%ASCII '#',    ! ASCII representation of a pounds sign
ASC_QUOTE       =%ASCII '''',   ! ASCII representation of a quote character
ASC_SPACE       =%ASCII ' ',    ! ASCII representation of a space
ASC_SQ_CLO_BRAK =%ASCII ']',    ! ASCII representation of a closed square bracket
ASC_SQ_OPN_BRAK =%ASCII '[',    ! ASCII representation of an open square bracket
ASC_TAB         = 9,            ! ASCII representation of a tab
ASC_UP_ARROW    =%ASCII '^',    ! ASCII representation of an up arrow

! The 'mode' data structure is really just
! a byte vector with the following characteristics.
!
MODE_LVL_SIZE   =10,            ! Number of bytes in each level.
MODE_LEVELS     =4,             ! Number of levels for mode settings

! Each level of the mode data strucutre has the following entries:
!
MODE_RADIX      =0,             ! Radix - dec, hex, oct, etc.
MODE_LENGTH     =1,             ! Length - long, word, byte, etc.
MODE_SYMBOLS    =2,             ! Boolean -> whether we know values
                                !     as "extern + offset" or not.
MODE_INSTRUC    =3,             ! Boolean -> whether we input/output
                                !     values as machine instruction.
MODE_ASCII      =4,             ! Boolean -> whether we output (only!)
                                !     values as ASCII strings or not.
MODE_SCOPE      =5,             ! Whether or not there is a CSP,
                                !     (and whether we should apply it)
MODE_GLOBALS    =6,             ! Whether or not we should apply global
                                !     scope first in the search rules.
MODE_FORTRAN    =7,             ! %LINE, %LABEL
MODE_IMMEDIATE  =8,             ! FORTRAN address or contents of mode
MODE_G_FLOATS   = 9,            ! Set the mode to G_FLOAT, so we can
                                !     pick up the constant as G_FLOAT
                                !     constant

! The four levels have the following names and indices.
!
DEFAULT_MODE    =0,             ! Default system initialized mode
USER_DEF_MODE   =1,             ! User-set default mode
OVERRIDE_MODE   =2,             ! One-line override mode
LOCAL_MODE      =3,             ! Local mode

! The mode_length field should be one of the following.
!
BYTE_LENGTH     =1,             ! Byte length
```

```
        WORD_LENGTH     =2,             ! Word length
        LONG_LENGTH     =4,             ! Longword length

        ! And the mode_radix field should be one of:
        !
        DECIMAL_RADIX   =10,            ! Decimal radix
        HEX_RADIX       =16,            ! Hexadecimal radix
        OCTAL_RADIX     =8,             ! Octal radix
        BINARY_RADIX    =2,             ! Binary radix

        ! The FORTRAN mode fields are as follows:
        !
        LABEL_MODE      =1,             ! Numbers are label numbers
        LINE_MODE       =2,             ! Numbers are line numbers
        LITERAL_MODE    =3,             ! Numbers are literal values


        ! The 'STEP' data structure is really just a byte vector with the
        ! following characteristics. It is isomorphic to the 'mode' structure.
        !
        STEP_LVL_SIZE   =4,             ! Number of bytes in each level.
        STEP_LEVELS     =3,             ! Number of levels for step settings


        ! Each level of the step data strucutre has the following entries:
        !
        STEP_LINE       =0,             ! LINE or INSTRUCTION (a boolean)
        STEP_NOSYSTEM   =1,             ! [NO]SYSTEM steps (a boolean)
        STEP_OVER       =2,             ! INTO or OVER (a boolean)
        STEP_SOURCE     =3,             ! SOURCE or NOSOURCE

        ! The three levels have the following names and indices.
        !
        DEFAULT_STEP    =0,             ! Default system initialized step
        USER_DEF_STEP   =1,             ! User-set default step type
        OVERRIDE_STEP   =2,             ! One-line override step type


        ! The data structure which holds the settings for SET SEARCH is
        ! a byte vector with the following characteristics (it works the
        ! same as the STEP data structure above)
        !
        SEARCH_LVL_SIZE = 2,
        SEARCH_LEVELS = 3,


        ! Each level of the data structure has the following entries
        !
        SEARCH_ALL = 0,
        SEARCH_IDENT = 1,


        ! The three levels have the following names and indices
        !
        DEFAULT_SEARCH = 0,
        USER_DEF_SEARCH = 1,
```

```
        OVERRIDE_SEARCH = 2,

        ! The data structure which holds the settings for SET DEFINE is
        ! a byte vector with the following characteristics (it works the
        ! same as the STEP data structure above)
        !
        DEFINE_LVL_SIZE = 1,
        DEFINE_LEVELS = 3,

        ! Each level of the data structure has only one entry
        !
        DEFINE_ONLY = 0,

        ! The three levels have the following names and indices
        !
        DEFAULT_DEFINE = 0,
        USER_DEF_DEFINE = 1,
        OVERRIDE_DEFINE = 2,

        ! The output configuration data structure is comparable to tne STEP or
        ! MODE data structures except that it has only one level.  It is used
        ! to store information pertaining to DEBUG's output setting.
        !
        OUTPUT_SIZE = 3,                ! Number of bytes in the data structure

        ! The following entries are contained in the output config vector:
        !
        OUT_LOG         = 0,            ! LOG or NOLOG
        OUT_TERM        = 1,            ! TERMINAL or NOTERMINAL
        OUT_VERIFY      = 2,            ! VERIFY or NOVERIFY
        OUT_SCREEN      = 3,            ! SCREEN_LOG or NOSCREEN_LOG

        ! Used to identify the flavor of type between dbg$show_type and its
        ! callers.
        !
        DEFAULT         = 0,
        OVERRIDE        = 1,

        ! Bit configurations for context flags.
        !
        CONTEXT_BITS    =32,            ! Number of context bits

        ! Position of opcode byte at a breakpoint
        !
        OPCODE_BYTE     =0,             ! Opcode offset

        ! Location types for end range arguments
```

```
        !
        MEMORY_LOC      =0,             ! Memory location

        ! Names of exception types for traceback.
        !
        NOT_AN_EXC      =0,             ! Line number searching for pc
        TRAP_EXC        =1,             ! PC of trap searching for line number
        FAULT_EXC       =2,             ! PC of fault searching for line number
        LOOKUP_EXC      =3;             ! Like TRAP only don't do VAL_TO_SYM
                                        !      again.

        ! Literals to define the various address spaces in the VAX architecture.
        !
LITERAL
        SYSTEM_SPACE    = %X'80000000',
        P1_SPACE        = %X'40000000';

LITERAL
        ! Language names
        !
        LANG_MACRO      = 0,
        LANG_FORTRAN    = 1,
        LANG_BLISS      = 2,
        LANG_COBOL      = 3,            ! COBOL
        LANG_BASIC      = 4,            ! BASIC+2
        LANG_PLI        = 5,            ! PL/I
        LANG_PASCAL     = 6,            ! Pascal
        LANG_C          = 7,            ! C
        MAX_LANGUAGE    = 6;            ! Languages 0 - 6 (C not supported yet)


        ! The semantic stack is a BLOCKVECTOR for which the following fields
        ! are defined...
        !
FIELD STACK_FIELDS =
        SET
        STK$V_BASE    = [0,0,0,0],      ! Base of block.
        STK$V_VAL1    = [0,0,32,0],     ! First value field
        STK$V_VAL2    = [1,0,32,0],     ! Second value field
        STK$V_NT_PTR  = [2,0,32,0],     ! Pointer to name table entry
        STK$V_INDEX   = [3,0,16,0],     ! Index field
        STK$V_OFFSET  = [3,16,16,0],    ! Offset field
        STK$V_POS     = [4,0,16,0],     ! Position field
        STK$V_TYPE    = [4,16,8,0],     ! Token type field
        STK$V_SIZE    = [4,24,8,0],     ! Size field
        STK$V_EXT     = [5,0,1,0],      ! Sign extension bit
        STK$V_REF     = [5,1,1,0],      ! Is token a REF bit
        STK$V_IMMED   = [5,2,1,0],      ! Immediate mode bit
        STK$V_ARGS    = [5,3,1,0],      ! Access actuals supplied bit
        STK$V_FLDRF   = [5,4,2,0],      ! <field_ref> supplied flag
        STK$V_DOT     = [5,6,1,0],      ! Indirection flag
        STK$V_STRUC   = [5,8,4,0]       ! Kind of BLISS structure
        TES;
```

```
LITERAL
        STACK_ENTRY_SIZ = 6,            ! No of longwords in a semantic_stack BLOCK
        NO_STRUC        = 0,            ! BLISS structure types
        VECTR           = 1,
        BITVEC          = 2,
        BLOK            = 3,
        BLOCKVEC        = 4;

MACRO
        SEMANTIC_STACK = BLOCKVECTOR[20, STACK_ENTRY_SIZ] FIELD(STACK_FIELDS)%;


! General purpose structure for manipulation of singly linked lists.  Each cell
! contains two long-word fields. The first is a pointer to the next list link
! cell; the second is a pointer to the current element in the list.
!
LITERAL
        LIST_LINK_SIZE  =  8;           ! Bytes in a list link cell

FIELD LIST_LINK_FLDS =
        SET
        NEXT_ONE_PTR    =  [0,0,32,0],  ! Pointer to the next link cell
        THIS_ONE_PTR    =  [4,0,32,0]   ! Pointer to the current object in the list
        TES;

MACRO
        LIST_LINK = BLOCK[LIST_LINK_SIZE, BYTE] FIELD(LIST_LINK_FLDS)%;
!
! End of DBGGEN.REQ
```

S0DEBUG
MAP

DBGLIB
REQ

DEBUG

DEBUG
MAP

DBGSSISHR
MAP

DBGEXT
REQ

DBGMSG
MDL

DBGGEN
REQ