M 11

```
DDDDDDDDDDDD        CCCCCCCCCCC   LLL
DDDDDDDDDDDD        CCCCCCCCCCC   LLL
DDDDDDDDDDDD        CCCCCCCCCCC   LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDDDDDDDDDDD        CCCCCCCCCCC   LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCC   LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCC   LLLLLLLLLLLLLLL
```

E 16

```
**FILE**ID**CLITABDEF





            CCCCCCCC  LL            IIIIII    TTTTTTTTTT    AAAAAA    BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
            CCCCCCCC  LL            IIIIII    TTTTTTTTTT    AAAAAA    BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
          CC          LL              II          TT      AA      AA  BB      BB  DD      DD  EE          FF
          CC          LL              II          TT      AA      AA  BB      BB  DD      DD  EE          FF
          CC          LL              II          TT      AA      AA  BB      BB  DD      DD  EE          FF
          CC          LL              II          TT      AA      AA  BB      BB  DD      DD  EE          FF
          CC          LL              II          TT      AA      AA  BBBBBBBB    DD      DD  EEEEEEEE    FFFFFFFF
          CC          LL              II          TT      AA      AA  BBBBBBBB    DD      DD  EEEEEEEE    FFFFFFFF
          CC          LL              II          TT      AAAAAAAAAA  BB      BB  DD      DD  EE          FF
          CC          LL              II          TT      AAAAAAAAAA  BB      BB  DD      DD  EE          FF
          CC          LL              II          TT      AA      AA  BB      BB  DD      DD  EE          FF              ....
          CC          LL              II          TT      AA      AA  BB      BB  DD      DD  EE          FF              ....
            CCCCCCCC  LLLLLLLLLL    IIIIII        TT      AA      AA  BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FF              ....
            CCCCCCCC  LLLLLLLLLL    IIIIII        TT      AA      AA  BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FF              ....


            SSSSSSSS  DDDDDDDD    LL
            SSSSSSSS  DDDDDDDD    LL
          SS          DD      DD  LL
          SS          DD      DD  LL
          SS          DD      DD  LL
          SS          DD      DD  LL
            SSSSSS    DD      DD  LL
            SSSSSS    DD      DD  LL
                  SS  DD      DD  LL
                  SS  DD      DD  LL
                  SS  DD      DD  LL
                  SS  DD      DD  LL
          SSSSSSSS    DDDDDDDD    LLLLLLLLLL
          SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

```
        MODULE $clitab;
/*      IDENT V04-000
```

```
/*++
/* Facility:    Command Language Interpreters, CLI Table Definitions
/*
/* Abstract:    This file contains the definitions for the data blocks which
/*              appear in a CLI table.  A CLI table is used by DCL and MCR
/*              to parse and execute DCL commands entered by the user.
/*              A CLI table is created with the Command Definition Utility.
/*
/* Environment: No assumptions can be made about the environment.
/*
/* Author:      Paul C. Anagnostopoulos
/* Creation:    7 December 1982
/*
/* Modifications:
/*
/*      V04-001 PCG0001         Peter George            06-Dec-1983
/*              Add NEG operator.
/*--
```

```
/*      C L I   T A B L E   B L O C K S
/*      -----   ---------   -----------

/* A CLI table contains all of the information that DCL and MCR need
/* to parse DCL commands.  The tables are composed of a set of blocks,
/* each of which describes one or more command items.  This SDL file
/* defines all of the blocks.
/*
/* Each block begins with a standard header, which is formatted as follows:
/*
/*      +-------+-------+---------------+
/*      |subtype| type  |    length     |
/*      |-------+-------|---------------|
/*      |   TRO count   |     flags     |
/*      +---------------+---------------+
/*
/* All references to other blocks are made via Table-Relative Offsets (TRO).
/* The TRO count specifies how many such references there are, and the
/* reference longwords always follow the header immediately.  The rest of
/* each block contains other information necessary for the definition of
/* the item.  Following the fixed portion of the block is a variable
/* portion, which contains any variable-length strings.  Each of these
/* strings is referenced from the fixed portion of the block by a
/* word Block-Relative Offset (BRO).
/*
/* The following list defines all of the valid block types.

constant (
        vector,                         /* Vector (root) block.
        command,                        /* Command block.
        type,                           /* Type block.
        entity,                         /* Entity block.
        expression,                     /* Expression block.
        cdu_visited                     /* For CDU internal use.
) equals 1 increment 1 prefix block_;

/* The following pages define the various block formats.  Many of the field
/* names are wierd, but have been retained for compatibility with previous
/* block formats.
/*      V E C T O R   B L O C K S
/*      -----------   -----------

/* The primary vector block appears at the beginning of a CLI table,
/* and contains references to all other blocks and block lists.

aggregate vector_block structure prefix vec_ fill;

size word unsigned;                             /* All blocks have a standard
type byte unsigned;                             /* header which is defined
subtype byte unsigned;                          /* up above.
        constant (
                dcl,                            /* Primary vector for DCL.
                mcr,                            /* Primary vector for MCR.
                verb,                           /* Verb name table.
                command                         /* Command block table.
        ) equals 1 increment 1;
flags structure word unsigned;
        strlvl byte unsigned;                   /* Format level of table.
        constant strlvl equals 6;               /* Current level is 6.
end flags;
tro_count word unsigned;
constant header_length equals .;

verbtbl longword;                               /* TRO of verb name table.
comdptr longword;                               /* TRO of command block pointer
                                                /* table.

table_size longword unsigned;                   /* Overall size of CLI table.

constant "length" equals .;                     /* Length of fixed portion.
end vector_block;


/* The verb name table is composed of the standard header, followed by one
/* longword for each verb or synonym.  The longword contains the first
/* four characters of the verb name, padded with NULs if necessary.

/* The command block pointer table is composed of the standard header,
/* followed by one longword for each entry in the verb table.  This
/* longword contains the TRO of the corresponding command block.
/*      C O M M A N D   B L O C K
/*      -------------   ---------

/* A command block is used to define a verb or a syntax change brought about
/* by a parameter or qualifier.  There is one command block for each verb
/* (but not for its synonyms), and one for each syntax change within a verb.

aggregate command_block structure prefix cmd_ fill;

size word unsigned;                             /* All blocks have a standard
type byte unsigned;                             /* header which is defined
subtype byte unsigned;                          /* up above.
        constant (
                verb,                           /* Verb definition.
                syntax                          /* Syntax change definition.
        ) equals 1 increment 1;
flags structure word unsigned;
        abbrev bitfield mask;                   /* Verb may be abbreviated
                                                /* non-uniquely.
        nostat bitfield mask;                   /* Command does not return a
                                                /* status, so don't check it.
        foreign bitfield mask;                  /* Command requests unparsed
                                                /* command line.
        immed bitfield mask;                    /* Immediate command, uses
                                                /* internal parsing routines.
        mcrparse bitfield mask;                 /* MCR style (output=input).
        parms bitfield mask;                    /* Parameter info is relevent.
        quals bitfield mask;                    /* Qualifier info is relevent.
        disallows bitfield mask;                /* Disallow info is relevent.
end flags;
tro_count word unsigned;
parms longword;                                 /* TRO of first parameter
                                                /* entity block.
constant max_parms equals 8;                    /* Maximum parameters.
quals longword;                                 /* TRO of first qualifier.
disallow longword;                              /* TRO of top-level disallow
                                                /* boolean expression block.

handler byte unsigned;                          /* How does CLI handle command?
        constant (
                none,                           /* It doesn't.
                cli,                            /* Calls a CLI routine.
                user,                           /* Calls a user routine.
                image,                          /* Invokes an image.
                same                            /* For syntax change, same
        ) equals 0 increment 1;                 /* handling as verb.
parmcnt structure byte unsigned;
        minparm bitfield length 4;              /* Minimum required parameters.
        maxparm bitfield length 4;              /* Maximum allowed parameters.
end parmcnt;
verbtyp byte unsigned;                          /* Verb type code for use with
pad1 byte fill;                                 /* old CLI interface.

name word unsigned;                             /* BRO of verb or syntax name.
image word unsigned;                            /* BRO of routine or image
                                                /* reference.
```

```
outputs word unsigned;                          /* BRO of outputs list.
"prefix" word unsigned;                         /* BRO of prefix string.

constant "length" equals .;                     /* Length of fixed portion.

variable character length 0 tag z;              /* Beginning of variable part.
constant max_name equals 1+31;                  /* Maximum sizes of variable
constant max_image equals 64;                   /* portions.
constant max_outputs equals 1+7;
constant max_prefix equals 1+31;
end command_block;

/* Following the fixed portion, the verb name(s) are stored as a sequence
/* of ASCIC strings within an overall ASCIC string.  Or, the syntax name is
/* stored as a single ASCIC string.
/*
/* The routine or image reference is stored as follows:
/*      CLI routine     Routine name as ASCIC string.
/*      user routine    Longword routine address, then name as ASCIC string.
/*      image           Image specification as ASCIC string.
/*
/* The outputs list consists of a counted sequence of bytes.  Each byte
/* contains either the negative of the parameter number, or the qualifier
/* number.
/*
/* The prefix string is stored as an ASCIC string.
/*      T Y P E   B L O C K
/*      -------   ---------

/* A type block is used as the header of a chain of entity blocks that
/* describe type keywords.  There is one type block for each user-specified
/* type definition.

aggregate type_block structure prefix type_ fill;

size word unsigned;                                /* All blocks have a standard
type byte unsigned;                                /* header which is defined
subtype byte unsigned;                             /* up above.
        constant (
                type                               /* Only one kind of type block.
        ) equals 1 increment 1;
flags word unsigned;
tro_count word unsigned;
keywords longword;                                 /* TRO of first keyword
                                                   /* entity block.

name word unsigned;                                /* BRO of type name.
"prefix" word unsigned;                            /* BRO of prefix string.

constant "length" equals .;                        /* Length of fixed portion.

variable character length 0 tag z;                 /* Beginning of variable part.
constant max_name equals 1+31;                     /* Maximum sizes of variable
constant max_prefix equals 1+31;                   /* portions.
end type_block;

/* Folowing the fixed portion, the type name is stored as an ASCIC string.
/* So is the prefix string.
/*      E N T I T Y   B L O C K
/*      -----------   ---------

/* An entity block is used to define each parameter, qualifier, and data
/* type keyword.  These blocks are linked off of the command block for
/* the verb, in the case of parameters and qualifiers, or off of a type
/* block, in the case of type keywords.

aggregate entity_block structure prefix ent_ fill;

size word unsigned;                             /* All blocks have a standard
type byte unsigned;                             /* header which is defined
subtype byte unsigned;                          /* up above.
        constant (
                parameter,                      /* Parameter definition.
                qualifier,                      /* Qualifier definition.
                keyword                         /* Keyword definition.
        ) equals 1 increment 1;
flags structure word unsigned;
        val bitfield mask;                      /* Can take a value.
        neg bitfield mask;                      /* Can be negated with "NO".
        deftrue bitfield mask;                  /* Present by default.
        batdef bitfield mask;                   /* Present by default if batch.
        valreq bitfield mask;                   /* A value is required.
        list bitfield mask;                     /* Can be a list of values.
        concat bitfield mask;                   /* Can be concatenated list.
        impcat bitfield mask;                   /* Implicit concantenated list
                                                /* (old CLI interface only).
        verb bitfield mask;                     /* Global placement.
        parm bitfield mask;                     /* Local placement.
                                                /* Both means positional.
        mcroptdelim bitfield mask;              /* MCR SET UIC kludge.
        mcrignore bitfield mask;                /* MCR ignores this entity.
end flags;
tro_count word unsigned;
next longword;                                  /* TRO of next entity block
                                                /* in chain.
syntax longword;                                /* TRO of syntax change
                                                /* command block.
user_type longword;                             /* TRO of type block for
                                                /* user-defined type.

number byte unsigned;                           /* Entity number.  CLI should
                                                /* only use for parameters.
valtype byte unsigned;                          /* Value type.
        constant (
                user_defined,                   /* Defined by user.
                infile,                         /* Input file spec.
                outfile,                        /* Output file spec.
                number,                         /* Decimal integer.
                privilege,                      /* Privilege keyword.
                datetime,                       /* Date/time.
                protection,                     /* Protection spec.
                process,                        /* Process name.
                inlog,                          /* Input logical name.
                outlog,                         /* Output logical name.
                insym,                          /* Input symbol name.
                outsym,                         /* Output symbol name.
                node,                           /* DECnet node spec.
                device,                         /* Node/device spec.
                dir,                            /* Node/device/directory spec.
                uic,                            /* UIC spec.
                restofline,                     /* Rest of command line.
                parenvalue,                     /* Parenthesized value.
                deltatime,                      /* Delta time only.
                quotedstring,                   /* String, and retain quotes.
                file,                           /* Any file spec.
                expression,                     /* General DCL expression.
                test1,                          /* Three hooks for testing
                test2,                          /* new data types before
                test3,                          /* adding them officially.
                acl                             /* ACL spec.
        ) equals 0 increment 1 counter #max,
        max_valtype equals #max;

name word unsigned;                             /* BRO of entity name.
label word unsigned;                            /* BRO of label used to
                                                /* retrieve entity.
prompt word unsigned;                           /* BRO of parameter prompt.
defval word unsigned;                           /* BRO of default value(s).

constant "length" equals .;                     /* Length of fixed portion.

variable character length 0 tag z;             /* Beginning of variable part.
constant max_name equals 1+31;                  /* Maximum sizes of variable
constant max_label equals 1+31;                 /* portions.
constant max_prompt equals 1+31;
constant max_defval equals 1+95;
end entity_block;

/* The entity name, label, prompt, and default values appear after the fixed
/* portion of the entity block.  They are stored as ASCIC strings.  The
/* default values are stored as a sequence of ASCIC strings within the
/* overall ASCIC string.
/*      E X P R E S S I O N   B L O C K
/*      -------------------   ---------

/* An expression block is used to represent, within a boolean expression,
/* one operator and its operands.  The operands are themselves expression
/* blocks, either subexpressions or paths.  Paths represent the hierarchical
/* path to an entity whose presence is to be determined.

aggregate expression_block structure prefix exp_ fill;

size word unsigned;                             /* All blocks have a standard
type byte unsigned;                             /* header which is defined
subtype byte unsigned;                          /* up above.
        constant (
                path,                           /* Entity path.
                not,                            /* Boolean NOT operator.
                any2,                           /* Boolean ANY2 function.
                and,                            /* Boolean AND operator.
                or,                             /* Boolean OR operator.
                xor,                            /* Boolean XOR operator.
                neg                             /* Boolean NEG operator.
        ) equals 1 increment 1;
flags word unsigned;
tro_count word unsigned;

constant "length" equals .;                     /* Length of fixed portion.

operand_list character length 0 tag l;          /* A TRO for each of the
                                                /* operands or path entities.
constant max_path_entities equals 8;            /* Maximum number of entities
                                                /* in a path.

end expression_block;

END_MODULE;
```

0068 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CRFOR
LIS

CRFMEM
LIS

CRFMACROS
MAR

CLITABDEF
SDL

FILINPUT
LIS

KEYS
LIS

CRFTFRVEC
LIS

DCLDEF
MDL

CRFMDL
MDL

CREF
LIS

CRFERRMSG
LIS

CRF
SDL

CRF

CRFSUB
LIS

FILOUTPUT
LIS

CRFSHR
MAP

CRFGLOBAL
LIS

CRFMAC
REQ

DCL

DCL
MAP

0069 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INTDEF
SDL

COMMAND
LIS

DESCRVAL
LIS

CONVERT
LIS

CLIMAC
MAR

CLIMSG
LIS

CONNECT
LIS

DCLPARSE
LIS

CHARMANIP
LIS

EXAMDEP
LIS

EXIT
LIS

INTIMAGES
MAR

DCXSTART
LIS

CLIGBL
LIS

DISALLOW
LIS

CLINT
LIS

CANCEL
LIS