A Macro File Syntax

A DABTool macro file consists of a sequence of macro definitions. Each macro definition consists of a header (macro name followed by a colon) and the body of the macro, a sequence of macro commands. A command consists of an identifier (name of macro or built-in command) and a parameter list (which must be empty in case of a macro call, i.e., macros with parameters are not supported; only the built-in operations may have arguments). An identifier consists of a letter followed by a sequence of letters, digits and the symbol '-'. Parameters may be integer, floating point, two-dimensional point and double-quoted string constants, denoted in the usual syntax (inside strings, the common C-like escapes for special characters like '`\n`' are also supported; points are denoted by pairs of floating point numbers enclosed in parentheses). The macro languages also supports variables: one can set any identifier to a value by means of the special '`set`' command (see below) and then later use this identifier as a command parameter.

The macro language is line-oriented. That is, each header and command must be on a line by itself. Blank lines are ignored. Line-oriented comments are denoted with the '`#`' character which causes all text up to the end of the line to be skipped. Continuation lines can be specified as usual using the backslash.

As already noted, most macro commands directly correspond to menu commands. Special commands of the macro language are '`set`' (define variable), '`unset`' (undefine variable), '`inc`' and '`dec`' (increment and decrement a numeric variable), '`message`', '`pause`', '`prompt`' and '`input`', which provide a means to interact with the user while a macro is running, '`repeat`' and '`end`' (simple looping construct), '`break`' (invoke debugger) and '`stop`' (stop macro execution). A complete list of the built-in macro commands which are currently available can be found below.

```
# macs - The DAB Tool Macro Language

### syntax:

# comments:
# # any text up to end of line ...

# macro definition:
# ID:
# ...

# statement:
# ID args ...

### parameters:
# ID - identifier
# X - any value
# p - point
# v, w - node number (>= 0: absolute; < 0: node from stack)
# n, m - integer
```