Debugger internal variables In addition to the file and subroutine-related variables mentioned above, the debugger also maintains various magical internal variables.

- `@DB::dbline` is an alias for `@{"::_<current_file"}`, which holds the lines of the currently-selected file (compiled by Perl), either explicitly chosen with the debugger's `f` command, or implicitly by flow of execution.

  Values in this array are magical in numeric context: they compare equal to zero only if the line is not breakable.

- `%DB::dbline`, is an alias for `%{"::_<current_file"}`, which contains breakpoints and actions keyed by line number in the currently-selected file, either explicitly chosen with the debugger's `f` command, or implicitly by flow of execution.

  As previously noted, individual entries (as opposed to the whole hash) are settable. Perl only cares about Boolean true here, although the values used by *perl5db.pl* have the form `"$break_condition\0$action"`.

Debugger customization functions

Some functions are provided to simplify customization.

- See Options in *perldebug* for description of options parsed by `DB::parse_options(string)` parses debugger options; see Options in *pperldebug* for a description of options recognized.
- `DB::dump_trace(skip[,count])` skips the specified number of frames and returns a list containing information about the calling frames (all of them, if `count` is missing). Each entry is reference to a hash with keys `context` (either `.`, `$`, or `@`), `sub` (subroutine name, or info about `eval`), `args` (`undef` or a reference to an array), `file`, and `line`.
- `DB::print_trace(FH, skip[, count[, short]])` prints formatted info about caller frames. The last two functions may be convenient as arguments to <, << commands.

Note that any variables and functions that are not documented in this manpages (or in *perldebug*) are considered for internal use only, and as such are subject to change without notice.

66.3 Frame Listing Output Examples

The `frame` option can be used to control the output of frame information. For example, contrast this expression trace:

```
$ perl -de 42
Stack dump during die enabled outside of evals.

Loading DB routines from perl5db.pl patch level 0.94
Emacs support available.

Enter h or 'h h' for help.

main::(-e:1):   0
  DB<1> sub foo { 14 }

  DB<2> sub bar { 3 }

  DB<3> t print foo() * bar()
main::((eval 172):3):   print foo() + bar();
main::foo((eval 168):2):
main::bar((eval 170):2):
42
```