

**NAME**

debug – set debugging level of diagnostic messages

**SYNOPSIS**

debug *level*

**DESCRIPTION**

The built-in global variable `DEBUG` is used to enable various levels of debugging print-out (spec must have been compiled with debugging enabled). You can use the `debug` macro to select a debugging level, or simply assign a value to `DEBUG`.

Add the following values together to obtain a debugging level:

1	0x1	Show input tokens during parsing.
2	0x2	Show node execution while running.
4	0x4	Show node allocate and free.
8	0x8	Show symbol table creation and lookup.
16	0x10	Show value get and set.
32	0x20	Show misc memory-related info.
64	0x40	Show hardware related info.
128	0x80	Show more hardware related info.
256	0x100	Show macro substitution.
512	0x200	Show memory allocate and free.
1024	0x400	Show input after macro expansion.
2048	0x800	Print warning messages.
4096	0x1000	Show certain low level hardware info.
8192	0x2000	Show data array allocation.
etc.	0x4000	Show signal blocking.
..	0x8000	Show low level hardware info.
.	0x10000	Show input file queuing.
.	0x20000	Show readable runtime debugging.
.	0x40000	Print input context on execution-time errors.
.	0x80000	Show sleeps.
.	0x100000	Show thread stuff.
.	0x200000	Show state changes.
.	0x400000	Use hexadecimal for socket debugging output.
.	0x800000	Show server/client socket messages.

Typing `+value` or `-value` adds or removes `value` from the current level.

Turning on debugging generates a large amount of output, so it is advisable to set the printer and data file to `/dev/null`. It is useful to save the debugging information in a *log* file. A typical debugging session to test what is going on with the hardware interfaces might be:

```
on("log")
debug 192
... commands to be debugged ...
close("log")
```

Alternatively, a special *dlog* file can be opened that will capture the debugging output, which will be sent nowhere else. For example:

```
fopen hdw.dlog
debug 192
... commands to be debugged ...
fclose hdw.dlog
```

Here the standard `fopen` and `fclose` macros are used instead of the built-in `on()` and `close()` functions.

**EXAMPLE**

```
debug +0x20000
```

**GLOBAL VARIABLES**

```
DEBUG
```

**SEE ALSO**

```
spec
```