Coding Guidelines for AOS

1 Code formatting

- 4 space indents, no hard tabs code formatted to 80 columns, mostly.
- Braces on the same line, unless it's a function, in which case the brace goes on a line by itself.
- Write C99 standard code, so // comments are ok.
- Macro names all upper case.
- Function names etc. all lower-case, use underscores to separate words.
- Space between if/for/while and the following parenthesis.
- Always use braces for if/for/while, even in the case of a single statement.

2 Coding construct guidelines

2.1 Types

- Never typedef a pointer or a struct/union/enum.
- All typedefs should be base integer types.
- Don't use the base types like int/long etc, use portable equivalents (eg. uint32_t, uint64_t, uintptr_t).
- Enum constants should be prefixed by the enum type name they belong to (and an underscore) and should be in camelcase.

2.2 Naming and Organisation

- Always mark functions and global variables static, unless they need to be accessed from code in another file.
- Name non-static functions with a common prefix related to the file name (eg. serial_putchar, microbench_register, etc.).
- Never put code in a header file, unless it's a static inline and needs to be there.
- Try to keep constants and preprocessor macros that are unique to a specific source file in that file rather than in a header that is included elsewhere.
  - Alternatively, create another header for the private declarations used just by that code.
- Keep definitions of structures and macros for accessing device hardware in a separate header file from any code/prototypes, and try to write them in a OS independent manner.

2.3 Return values

Use fugu for error values (see tools/fugu and lecture).

2.4 Misc

- Use doxygen comments.
- Use assert() liberally for sanity checks, but if something might fail at runtime, make sure it still behaves correctly (i.e., returns an error code) with asserts compiled out.

3 Example code

```
/**
 * \brief This function does nothing of any significant purpose
 *
 * \param arg Argument description
 * \return uintptr_t Return value description
 */
uintptr_t frongle_blarg(void *blah)
{
    if (blah) {
        return (uintptr_t)blah - PAGESIZE; // foo
    } else {
        assert(!"Badness!");
        return NULL;
    }
}
```