**CS 350: 4-2 Journal - Coding Best Practices**

Embedded C divides the best coding practices into several areas. Among these are white space rules, module rules, data type rules, process rules, variable rules, and statement rules.

Each article includes tips on how to prevent common Embedded C issues.

The white space guidelines are as follows. Variable and structural names, as well as assignment operators, must be properly aligned, and spaces must be properly inserted. To separate paragraphs and file ends, use blank lines. The indentation should be four characters from the start of the line. "The tab (ASCII 0x09) character shall never appear within any source code file" (Barr, 2018). The only non-printable characters in the source code should be 'LF' and 'FF' (Barr, 2018).

Module rules must adhere to proper name conventions and include header files, source files, and file templates. It is also necessary to have fixed-width integers, proper signed and unsigned integers, floating points, structures and unions, and Booleans. Procedure rules include naming conventions that are compatible with each other, functions, function-like macros, execution threads, and interrupt service routines. Variables must also be properly labeled and initialized. Finally, variable declarations, conditional statements, switch statements, loops, jumps, and equivalency tests must match the rest of the code (Barr, 2018).

There are some common pitfalls to avoid while programming with Embedded C. Inserting spaces where they do not belong or neglecting to place them where they do belong is a common problem that creates eyestrain, makes it harder to discover errors, makes it difficult to distinguish between different blocks of code, produces inconsistent layouts, and causes other issues. Another risk is the misuse of module rules, which can lead to inconsistent declarations, unnecessary coupling, and other errors. If a programmer disregards data type standards, it can lead to problems with code portability, data-dependent consequences, and the possibility of defects. Following data procedures eliminates distractions and hang-ups while also making debugging and assessment easier. Variable and statement rule consistency assists in "maximizing code portability across compilers," reducing security concerns, minimizing risks, and synchronizing data (Barr, 2018).

When working with Embedded C, it is critical to understand and follow correct coding practices to avoid code errors and security issues. It also makes the code easier for reviewers and other developers to read and follow. This, in turn, makes it easier to spot flaws, errors, and vulnerabilities in the code.

**References**

Barr, M. (2018).Embedded C Coding Standard. Barrgroup.com. Retrieved November 21, 2021, from

[https://www.coursehero.com/file/77005463/barr-c-coding-standard-2018pdf/.](https://www.coursehero.com/file/77005463/barr-c-coding-standard-2018pdf/)