Sprints

Sprints Coding Guidelines

Automotive BootCamp

Sprints

Table of Contents

General Rules

- **Include** all the needed files only in module header file.
- **Include** the module header file first, then any other header file that are not needed to be shared and needed in the source file logic.
- **Macro(s)** and Function like Macros shall be written in Capital Letters.
- **Enumeration** Values shall be written in Capital Letters.
- **API(s)** shall return status code. All the API return variables shall be passed by reference the corresponding API(s).
- **Check** for all the non-void API(s) return value. And continue your logic only over the correct return status code.
- **Propagate** any un-expected status code to the upper layer. In case there are no action to be taken in the current layer for such failures.
- **Use** Parenthesis **()** always in combined conditions.
- **Use** Curly Braces **{}** in loops, conditions, switch (and its cases). Even if the logic to execute is one line.
- **Insert** curly braces in new lines.
- **Use** break statement when searching for a specific element inside an array of elements.
- **Construct** all the global variables of the module inside its initialization API.
- **Destroy** all the global variables of the module inside its de-initialization API.
- **Put** all the local variables inside the local function after its curly braces.
- **Initialize** all the local variables before using them.
- **Cover** all the code paths, even if it's not used (Implement it and leave it blank with "Do Nothing" Comment.
- **Check** for the pointer value and assure it's not equal to null before accessing the pointer. Same behavior needs to be considered over pointer to functions and callback functions.

- **Validate** the function input parameters and assure they have valid values before using them in your logic or database.
- **Manipulate** global variables with setter and getter macro like functions or real functions (API(s) or inline static functions. Depending on the usage scope of setters and getters)
- **Avoid** using recursive functions. Instead, put your logic inside a while loop and make your recursive function base case the condition you are looping until it evaluates to false.
- **Avoid** Using Primitive Types Directly. Instead, implement your own types and use them.
- **Use** static keyword with all module's global variables "Not Shared Variables".
- **Use** static keyword with all module's helper functions (Functions that are not API(s).
- **Assign** The first Enumeration value to (0) or to any other proper value.
- **Indent** all the statements inside curly braces with one tab (4 Spaces).
- **Indent** any conditional preprocessor with one tab depending on their depth of nesting.
- **Put** the extern keyword with the exported variable inside the header file of the module. If it's needed to be shared with all the modules that will use it.
- **Add** a comment describing the logic before the start of the logic chunk.
- **Implement** your own secured version of malloc and free API(s). As the original API(s) are not secure from the scope of multiple pointer allocations and null pointer deletion.

General Mistakes to Avoid

- **Never** include source code file (.c) into another source code file (.c) or in a header file (.h) “This is even worse”.
- **Never** do any bitwise operation for any signed variable.
- **Never** use the size-of operator in looping over array elements. Instead Implement a Macro to determine the array length and use it.
- **Never** use the array length macro in any send API over a communication medium. Same with the Receive Callback. Instead, use the size-of operator.
- **Never** put the variable first in the condition, then its value. Instead, put the variable’s value to compare first (in the left operand), then the variable name in the right operand.
- **Avoid** returning the address of a local variable inside a function.
- **Avoid** using a de-allocated pointer.
- **Never** Do Circular Inclusion. As the compiler will not include the last file to terminate the infinite loop of inclusion. And this may produce warning(s).

Sprints

Naming Convention

- **Type Definition Names** shall illustrate their types and usage**.**

Type Definition Naming Convention

- **Primitive Type Definition** shall be prefixed with **u (in case of unsigned) || s (in case of signed).** Then, joined with **intx** (where x is the size in bits). Also, it shall be postfixed with **t**_
- **Pointers** shall be prefixed with **ptr**. Also, shall be postfixed with **t**_
- **Pointer to Functions** shall be prefixed with **ptr_func.** Also, shall be postfixed with **t**_
- **Structure Type Definition** shall be prefixed with **str** and postfixed with **t**_
- **Structure Type Definition Tag** shall be prefixed with __ and shall be assigned the same **Structure Type Definition** (Same as overloading naming convention in C++).
- **Enumeration Type Definition** shall be prefixed with **enu** and postfixed with **t**_
- **Enumeration Type Definition Tag** shall be prefixed with __ and shall be assigned the same **Enumeration Type Definition** (Same as overloading naming convention in C++).
- **Union Type Definition** shall be prefixed with **uni** and postfixed with **t**_
- **Union Type Definition Tag** shall be prefixed with __ and shall be assigned the same **Union Type Definition** (Same as overloading naming convention in C++).

Variables Naming Convention

- **Always** give your variables meaning names. The naming shall illustrate the scope, variable type and the variable usage.
- **All Variable Names shall illustrate their types as illustrated in the above section.**
- **Global** Variables shall be prefixed with **gl.**
- **Pointer** Variables shall be prefixed with **ptr.** (Except **Pointer to functions**).
- **Pointer to Functions** Variables shall be prefixed with **ptr_func.**
- **Constant** Variables shall be prefixed with **cst.**
- **Structure** Variables shall be prefixed with **str.**
- **Enumeration** Variables shall be prefixed with **enu.**
- **Union** Variables shall be prefixed with **uni.**