## Coding Guidelines

* FileNames should follow CamelCase naming convention. Eg:- SensorAdpdModule.c
* All variables should be explicitly initialized to zero rather than leaving it for the compiler
* Lines in file should be no more than 80 characters long (most editors have ruler that you can set to check if it crosses the limit), since this will have a large impact on formatting when code is to be used for documentation. Break long lines instead to fit within this character limit on a line. Block comments above the code block also should within 80 characters as well.
* If there are some changes to be done in future or fixed, then “TODO” should be used as the string comment. This will help to search later and make this change. Include NOTE as a string comment to leave an important message.
* Proper indentation should be used for better readability of code. Eg:- in the code block below, each block of code (for) is indented by 4 spaces from the enclosing block. **Avoid using tabs in code.**

int32\_t MyFunc(int32\_t nMyVar) {

for (i = 0; i < LOOP\_COUNT; i++) {

     code block

    }

}

1. Indent 'class' and 'struct' blocks
2. Indent 'switch' blocks so that the 'case X:' statements are indented in the switch block. The entire case block is indented.
3. Add extra indentation to labels so they appear 1 indent less than the current indentation, rather than being flushed to the left (the default)
4. Indent multi-line preprocessor definitions ending with a backslash
5. Indent preprocessor conditional statements to the same level as the source code

* To improve readability, symbol declarations and comments should be aligned

bool bFlag;               /\* this is a flag \*/  
ADI\_DEV\_HANDLE \*pHandle; /\* this is a pointer \*/  
ADI\_INT\_RESULT eResult;   /\* this is the result \*/

1. Delete empty lines within a function
2. Source file should not be too long. Default is 2000
3. Reserved names should not be used for preprocessor macros
4. Source files should contain the copyright notice
5. At least two spaces is best between code and comments

* Avoid extraneous whitespace in the following situations:
  + 1. Immediately after opening parentheses, brackets or before closing parentheses, brackets
    2. Immediately before the open parenthesis that starts the argument list of a function call
    3. Immediately before the open bracket that indicates an array indexing.
    4. More than one space around an assignment (or other) operator (for example, to align it with another operator on another line)
* All binary operators (e.g. ==, , !=, , <=, >=, =, +=, -= etc.) except . and -> shall always have a single space placed on either side of the operator.

1. Single spaces shall be used either side of the operators in the ? : ternary operation.
2. Spaces shall not be used after a unary operator (+, -, !, ~, \*, & etc)
3. Use a single space after a comma (,)
4. A single space shall be used after keywords that are followed by parenthesis (like if and for)
5. No line shall end in whitespace.
6. The "pointer" qualifier (\*) shall be associated with the variable name rather than the type. There shall be no whitespace after the \* character
7. Each expression in a for statement shall be separated by a space

* Prefixes that should be used:
  + 1. n  - Any numerical value nSamples,
    2. r  - Bounded range (value with min/max) rBound
    3. e  - Enumerated value (pre-defined values) eState, eEvent, eResult
    4. b  - Boolean value bFilterEnabled
    5. c  - Any character value (including widechar) cKeyPressed
    6. sz  - Zero-terminated string szMessage
    7. s  - Non-terminated string sErrorMsg
    8. h  - Handle (defined in Terminology) hInstance
    9. o  - Any structured object oComponent
    10. pf  - Pointer to function/method pfCallback
    11. Qualifiers
        - Pointer to... peCacheMode, ppNext, panSamples
        - Array of... aComponents
        - k  - Constant knMaxElements
        - g  - Global (file-scoped) ganLookupTable