

# Coding Standards

product information

## Coding Standards

By following these coding standards, we make our code easy to maintain and understand. It also allows for our code to be uniform, no matter which developer is working on it.

## Coding Rules

1. No tabs are to be used in the source code
2. All indentations must be to four (4) spaces to indicate scope
3. Line continuations are to be indented to more than eight (8) spaces as to differentiate the line continuation from scope levels
4. All braces "{}" must be on a single line with no other code and Not be indented at the same level as the contents inside.
5. Only one executable statement per line is to be used in any source file
6. Comments must be descriptive and not restate the obvious
7. No comment boxes are to be used.
8. Comments should be in column one (1) with the "/\*" and the wording in the comment should begin in column four (4)
9. Lines of source code should not exceed 80 columns
10. The use of negative operations should be avoided
11. The "C" language should not be extended through the use of macros (i.e. forever and unless)
12. Variable declarations must be aligned by variable name by placing the first alphabetic character in each variable name in the same column
13. Variable names must start with a lower case letter and use a capital letter for the first letter of each following word or abbreviation (i.e. localVariableName)
14. All files must include the Company copyright along with the description, author, filename, date created, and modification history at the beginning of the file
15. No underscores "\_" are to be used in variables, typedefs, or structures
16. Definition Macros must be in Upper Case with underscores "\_" following the prefix (if there is one) and underscores between each word or abbreviation
17. Code Macros must be in lower Case with underscores "\_" following the prefix (if there is one) and underscores between each word or abbreviation
18. Variable names must be chosen so that they are meaningful. If abbreviations are used, the meaning MUST be clear and unambiguous
19. All constants MUST be defined in a Macros
20. ANSI "C" compliance will be followed with full function prototypes



Mobile



Network

**Treck Inc.**

# Coding Standards

21. All project function prototypes will be located in a single header file
22. Global Variables will be defined in a single file for each project
23. The const type will be used for all static tables
24. "C" types should not be explicitly used except for compatibility issues
25. Header Files will not be nested
26. All return codes will be checked and dealt with accordingly
27. The default case in a switch statement will NEVER be left blank
28. All fall though case statements will be documented with comments as to why the fall through needs to occur
29. Error Checking should always be done on anything that might go awry (i.e. overflow, multithreaded access to pointers, etc.)
30. 32 bit and 16 bit values will always be fully aligned (even in linked lists)
31. A compile of the code should produce No Warnings.
32. Any variable declared must be used, unless it is declared as a parameter in state machine functions and is not used in every state function
33. Pointers are always declared with a pointer type not the asterisk (\*)

## Naming convention

34. Global Variables are prefixed with "tv" (i.e. tvSystemTicks)
35. Static/Local variables are prefixed with "tl" (i.e. tlCountVariable)
36. Typedef types are prefixed with "tt" (i.e. ttListType)
37. Function names are prefixed with "tf" (i.e. tfGetValue)
38. Structure names are prefixed with "ts" (i.e. tsListStruct)
39. Union names are prefixed with "tu" (i.e. tuParameterUnion)
40. Macros are prefixed with "TM\_" (i.e. TM\_MAX\_BYTES)
41. Macro code is prefixed with "tm\_" (i.e. tm\_lock)
42. Pointers are postfixed with "Ptr" (i.e. listPtr)

## Exception

The only exception to these naming conventions is when a specification or RFC requires otherwise.



Mobile



Network

