VxWorks Coding Standard Updates

Document ID:	VxWorksCodingStandardUpdates
Author:	Mati Sauks
Version:	0.01
Version Date:	Feb 1, 2019
Status:	Draft

Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): +1-800-545-WIND
Telephone: +1 510 748 4100
Facsimile: +1 510 749 2010

For additional contact information, see the Wind River website:
http://www.windriver.com

For information on how to contact Customer Support, see:
http://www.windriver.com/support

REVISION HISTORY			
Date	**Version**	**By**	**Description of Change**
01 Feb 2019	0.01	Mati Sauks	First Issue

Table of Contents

List of Figures

No table of figures entries found.

List of Tables

1 Introduction

1.1 Purpose and Scope

This document is to provide information about the changes from the VxWorks Coding Standard of Dec 5, 2014, and the new VxWorks Coding Style Guide and the VxWorks Coding Rules Guide documents.

1.2 Intended Audience

This document is for veteran VxWorks engineers who have knowledge of the previous coding standard.

1.3 References

The following documents and links are referenced in this document:

Ref.	Title
Ref.1	*VxWorksCodingStandard_Dec. 5, 2014*
Ref.2	*VxWorks Coding Style Guide Version 1.0*
Ref.3	*VxWorks Coding Rule Guide Version 1.0*

Table 1-1 Applicable Documents

2 The new coding guides

The original VxWorks Coding Standard document [Ref.1] has been split into two documents:

VxWorks Coding Style Guide [Ref.2]

VxWorks Coding Rules Guide [Ref.3]

2.1 VxWorks Coding Style Guide

The style guide documents the formatting that the code and the comments must follow.

2.2 VxWorks Coding Rules Guide

The rules guide documents the usage of the programming language that must be followed.

3 VxWorks Coding Style Guide

3.1 Doxygen and APIGEN

APIGEN has been deprecated, the use of Doxygen markup is now documented in the Style Guide (see the chapter Doxygen Description Language). All of the examples use the Doxygen markup.

3.2 Standard File Header (Section 3.2)

This section has been updated to provide more guidance on Modification History. Sections were added for DTS, .spec and .vsbl files.

3.3 C Function Layout (Section 4.2)

The description section of the function layout has been updated to require more information. This includes a full description of what the function is to do, and documentation of all input and output parameters on top of the return value. See the Parameters Lists subsection of the Doxygen chapter.

3.4 C Declaration Formats – Variables (Section 4.2.10)

The standard has been updated to specify that basic variable types should use the ones specified in stdint.h, for example uint32_t.

3.5 Functions and Variable Names (Section 4.4.5)

New naming for internal functions that are not static:

Internal functions and variables that are not static should follow the following convention, either starting with a double underscore with a non-capitalized letter, or a single underscore character followed by a capital letter. For example:

```
__myJimmyGet ();
_MyJimmyGet ();
```

3.6 Private, Protected , and Public Interfaces (Section 4.4.6)

A new section to provide clarification on these interfaces to layers.

3.7 extern keyword usage (Section 4.4.7)

Clarification on the usage of the extern keyword.

In a ‘C’ source file the extern keyword should only apply to static (local) variables and functions. All other extern statements should be in private/public header files.

All externs for a subsystem should be in the subsystem header files. Users of the subsystem should include the subsystem header file and NOT declare any subsystem externs in their own header file(s).

3.8 Assembler Coding Style (Chapter 5)

This is a new chapter.

3.9 C++ Coding Style (Chapter 7)

This is a new chapter.

3.10 Python Coding Style (Chapter 8)

This is a new chapter.

3.11 Doxygen Reference Standards (Chapter 10)

This replaces the API Reference Standards chapter (APIGEN).

3.12 API Control in VxWorks – Reference (Old Chapter 11)

The Reference section relating to release numbers was removed.

3.13 Versioning in VxWorks (Chapter 11)

This chapter was re-written to increase the clarity and use of VxWorks 7 terminology.

3.14 Deprecated Coding Standards

This chapter was removed.

4 VxWorks Coding Rules Guide

4.1 Coverity Defect Suppression (3.1)

New section. The section clarifies the use of coverity defect supression annotations.

4.2 Certified Code (3.2)

New section. This describes the use of certification Requirements tag artifacts in the source code:

/* req: XXXXXXX */

4.3 Standard Library Functions (3.3)

Updated the rules to indicate that the "n" version of functions should be used, i.e. use strncmp() instead of strcmp().

4.4 Use Defined Names (3.6)

The defines for IMPORT and LOCAL are no longer to be used.

4.5 Unused Return Values (3.10)

Updated slightly.

4.6 Signed Unsigned Comparisons (3.13)

New section. Mixing of signed and unsigned variables and constants can lead to unexpected consequences, especially when comparisons are being made.

4.7 Avoid Use of Casts (3.17) Type Casting (3.18)

Updates to casting section. If you do use a cast, make sure it is between appropriate types, for example ensure that the cast can never result in unintended information loss (such as casting a long to an int).

4.8 Function Parameters section updated to Side Effects (3.20)

Renamed section, added a few more examples of side effects to avoid.

4.9 For Loops (3.23)

More examples added with respect to usage of loop variables.

4.10 Recursion (3.25)

Example updated.

4.11 Automatic variables (3.26)

Added content dealing with eclipsing previous declarations in outer scope.

Sub-block scoped automatic variables are acceptable, with the proviso that the name of an automatic variable shall not eclipse (have the same name as) an automatic variable declared with function scope (or a global variable). Similarly an automatic variable declared with function scope shall not eclipse a global variable.

4.12 Processor and architecture specific code (3.27)

Update to enforce the fact that processor and architecture specific code should not be included in non-architecture specific files via the “asm” macro or #if /#ifdef macros and other macros.

4.13 Restrictions on keywords (3.28)

Renamed section, moved setjmp/longjmp to section 3.3, added more examples.

Previously “goto” was to be avoided. Now it is allowed for error handling paths.

4.14 The Ternary (?) Operator (3.29)

Use of ternary operator rules relaxed. It can be used, but should be avoide for complex operations.

4.15 Use of the static keyword (3.30)

New section. All functions and global variables used only in a compilation unit shall be declared static. Do not export symbols that should not be accessible.

4.16 Code Complexity (3.31)

New content. Avoid complex expressions. Complex expressions make it difficult to maintain and to certify.

4.17 Code Complexity renamed to Code Complexity measurement (3.32)

Extra information on how to measure complexity added.

4.18 Pointer parameters to public APIs renamed to Parameters of Public APIs (3.33)

Updated to include all parameters, plus improved example. All public APIs parameters should be validated for correctness when possible.

4.19 Global Variables (3.34)

Updated to add/remove statements for clarification to enforce the concept that global variables while allowed, should be used with care, with access functions where possible.

4.20 Return Value from ISRs (3.35)

Updates to the text to clarify how the return value from an ISR is used.

4.21 Misuse of the 'Register' Attribute (3.39)

Do not use the 'register' or 'FAST' keywords.

4.22 Avoid Vector Name

Section removed.

4.23 Statement Labels

Section removed.

4.24 VxWorks Macros (3.4.1)

New section to present __VXWORKS__ and __vxworks

4.25 Summary of Compiler Macros renamed to Toolchain Abstraction Macros (3.4.2)

All of the VxWorks macros and others are gathered together into this section and updated for clarity.

4.26 Complex Macros (3.4.3)

New section. Complex function like macros should be avoided. An inline function or preferably a regular function should be used in preference to a macro.

4.27 C++ Coding Rules (Chapter 4)

New chapter.

4.28 Assembler Coding Rules (Chapter 5)

New chapter.

4.29 Special Rules for Certified Code

This section has been removed and content embedded into the document as general rules where appropriate.