Overview of Wind River Workbench - C++test Integration

This topic provides basic information about the capabilities of the C++test Wind River Workbench plugin and explains how to install the C++test plugin into Wind River Workbench.

In this section:

Integration Overview

The C++test plugin is integrated with the Wind River Workbench IDE by means of the Eclipse plugin mechanism. C++test plugs into Wind River's IDE by adding a link file to the following location: `<Wind River Installation Root>/workbench-version_number>/wrwb/platform/eclipse/links/com.parasoft.xtest.cpptest.link`. This link file contains the path to the C++test plugin; this informs Workbench that the plugin should be loaded at startup.

After installation C++test will add its views to Workbench default "Application Development" perspective. Also, a separate C++test perspective will be available in Workbench.

Static Analysis Support

Static analysis (including coding standards and data flow [with the appropriate license options]) is supported for the following project types:

- VxWorks Downloadable Kernel Module
- VxWorks Real Time Process Project
- VxWorks Shared Library Project
- VxWorks 653 makefile project

Both build support strategies (Flexible and Standard) are equally supported. For more details, see Static Analysis with the C++test Wind River Workbench Plug-in.

Unit Testing Support

The complete unit testing flow is supported for:

- VxWorks Real Time Process Projects
- VxWorks Downloadable Kernel Module Project
- VxWorks Shared Library Project

Both build support strategies (Flexible and Standard) are supported. However, additional configuration may be required in some situations—for example, manually excluding generated test components (such as test cases and stubs) from the original build process, or generating additional test-required data. This will be covered in detail in later sections that describe the setup process for each working model. For more details, see Unit Testing with the C++test Wind River Workbench Plug-in.

General Information on Setting up Testable Projects

The Wind River tool suite works with numerous types of projects, depending on target platform and application type. To support testing different project types following approaches can be used:

- Test Wind River Workbench projects directly
- Test Wind River Workbench projects through test projects
- Custom setup (for VxWorks 653 projects)

Wind River VxWorks 653 projects requires a custom project setup procedure.

There are two possible approaches to testing source code located in Wind River Workbench projects (projects with either Flexible or Standard build support):

- Test Wind River Workbench projects directly
- Test them through "Test Projects".

When we talk about testing Wind River Workbench projects "directly," it is assumed that no additional entity to the workspace is added in order to enable testing such a projects. You select either the complete project or any subset of resources from this project and start the Test Configuration.

If it is not possible to test a Wind River Workbench projects directly, you need to create a test project and test the code through the test projects.

In general you can test Wind River Workbench projects directly when:

- The project to be tested does not have any sub projects.
- The project includes all the resources added to build targets of the project under test.

When your project does not satisfy these conditions, you should test your source code through a test project. For example, test projects are commonly needed for multi-layer projects with few levels of sub projects (like: VIP/DKM [containers]/DKM [source projects]) or projects that use flexible build support and have a build target definition that contains resources coming from other projects.

Known Problems and Limitations

Options Scanning Problems

When C++test starts the analysis, the first action it performs is scanning the make file for the compilation/linking options. To ensure that the testing session runs successfully, you need to ensure that all rules (compilation and linking) required to build the project will be fired for all required files. See Important Note - Forcing Make to Unconditionally Build All Targets for details on how to do this.

Limitations in Supporting Language Extensions

Unsupported compiler keywords

- extended
- __interrupt__ and interrupt
- __packed__ and packed
- bool, pixel, vec_step, and vector AltiVec Keywords
- __ev64_* Keywords
- __accum, __fixed, __X, and __Y DSP Extensions
- pascal

Limitations in handling intrinsic functions (Diab compiler)

C++test may fail parsing when the following intrinsic functions or type specifiers are used in the source code:

ARM:

- ffi

PPC:

- multiply-accumulate instructions (PPC405)
- AltiVec instructions group

Sh:

- __fixed type

Spar:

- __scan and __divscc

Other Limitations

- The C++test plugin for Wind River Workbench supports the command line interface only for testing "test projects". Direct testing of Wind River workbench projects is not supported.
- The C++test plugin for Wind River Workbench does not support unit testing for projects with Flexible build support when the resources added to build target definition do not belong to the same project as the build target definition.