As the software components in embedded systems are becoming increasingly critical, the attention to quality in embedded software increases across the board. Long-standing quality strategies such as testing with a debugger are no longer efficient or sufficient. To further complicate matters, many developers cannot readily run a test program in the actual deployment environment because they lack access to the final system hardware. To address these challenges, code quality needs to be realized throughout the development lifecycle—using a synergy of time-proven techniques for early defect prevention, assisted by automation for implementation and monitoring.

C++test from Parasoft enables teams to produce better code for embedded systems, test it more efficiently, and consistently monitor progress towards their quality goals. With C++test, critical time-proven best practices—such as static analysis, comprehensive code review, and unit and component testing with integrated coverage analysis—are enabled on the developer’s desktop, early in the development cycle. A command line interface enables fully automated execution within regression and continuous integration environments, providing data for monitoring and analyzing quality trends.

For highly quality-sensitive industries, such as avionics, medical, automobile, transportation, and industrial automation, the addition of Parasoft’s Web-based audit and reporting system, with interactive Web-based dashboards and drill-down capability powered by a SQL database, enables an efficient and auditable quality process with complete visibility into compliance efforts.

C++test Lauterbach TRACE32 Integration

Runtime testing techniques such as Unit Testing or Memory Error Detection are essential in development of high quality software—especially for safety critical applications. Building an effective and efficient C/C++ code runtime testing environment requires seamless integration between the development tools and testing tools.

With C++test’s support for the TRACE32 debugger, users can easily achieve full automation of the testing process. Activities like downloading the tested code to the target, test execution, and results acquisition are controlled by a C++test-generated PRACTICE language script, which is interpreted by Lauterbach’s debugger. The PRACTICE scripts that C++test generates for test execution automation can be easily modified or integrated with existing scripts (for example, for initializing the target device). This makes the solution easy to apply to both new and existing software projects.

In addition to the fully-automated and non-interactive testing mode, users can take full advantage of the Lauterbach debugger to step through test cases. The ability to debug test cases significantly speeds up the process of creating robust and reliable regression test suites. Moreover, C++test’s architecture allows users to easily switch between testing their code on real hardware and simulators provided with TRACE32—enabling teams to start testing early in the production phase, when the hardware platform may not yet be ready.

Unit and Integration Tests with Coverage Analysis

C++test’s automated testing helps establish correctness and reliability of newly-developed or legacy code. C++test automatically generates complete tests, including test drivers and test cases for individual functions, purely in C or C++ code in a format similar to CppUnit. Alternatively, users can interactively define tests using a Test Wizard. These tests, with or without modifications, are used for initial validation of the functional behavior of the code.
High Degree of Customization

C++test allows full customization of its test execution sequence. Users have control over the content of the PRACTICE script generated for the TRACE32 debugger that automates testing process. Besides the typical script sections for initializing the target device and setting up the FDX communication for tests results transport, users can easily utilize the power of the TRACE32 debugger by implementing test scenarios where, for example, a simulation of memory corruption is performed with the help of Lauterbach tools.

In addition to using the built-in test automation, users can incorporate custom test scripts and shell commands to fit the tool into their specific build and test environment. This unparalleled flexibility enables users to realize their desired test flow without being constrained by the preset tool options. C++test can be utilized with a wide variety of embedded OS and architectures, by cross-compiling the provided runtime library for a desired target runtime environment. All test artifacts of C++test are source code, and therefore completely portable.

Code compliance, data flow analysis and code review

Besides runtime testing techniques that are seamlessly integrated with TRACE32, C++test offers a unique combination of code structure and static data and control flow analysis. Users can easily implement coding policies that eliminate entire classes of programming errors and assure source code compliance with Industry standards like: MISRA, MISRA C++ 2008, JSF. To configure C++test to enforce a coding standards policy specific to their group or organization, users can define their own rule sets with built-in and custom rules.

The advanced interprocedural data and control flow analysis module of C++test simulates feasible application execution paths—which may cross multiple functions and files—and determines whether these paths could trigger specific categories of runtime bugs. Defects detected include using uninitialized or invalid memory, null pointer dereferencing, array and buffer overflows, division by zero, memory and resource leaks, and various flavors of dead code.

Code review is known to be the most effective approach to uncover logical errors. The C++test Code Review module automates preparation, notification, and tracking of peer code reviews. Status of all code reviews, including all comments by reviewers, is maintained and automatically distributed by the C++test infrastructure.

Extensive reporting

Code analysis reports can be generated in a variety of formats, including HTML, PDF and raw xml files which are suitable for further processing. Multitude of information included in reports and high degree of available customizations makes reporting module indispensable in common day work as well as during the audits and reviews.

Advanced Unit Test features:

- Automatic generation of tests and stubs
- Automatic generation of assertions based on observed test results
- Graphical Test Case Wizard for interactive definition of tests
- Complete visibility into test and stub source code
- Intelligent, test-case-sensitive stubs
- Parameterization of tests and stubs
- Multi-metric coverage analysis (including MC/DC)
- Flexible support for continuous regression testing
- Annotation of tests against bug and requirement IDs
- Execution of tests under debugger
- Special mode for testing template code

C++test’s customizable workflow allows users to test code as it’s developed, then use the same tests to validate functionality/reliability in target environments.