Product Name: VectorCAST/Probe

Features

- Captures internal data values
- Records detailed control flow
- Injects Faulty Values to test error handling
- Debugs hard to trigger race conditions
- Supports all embedded targets

Benefits

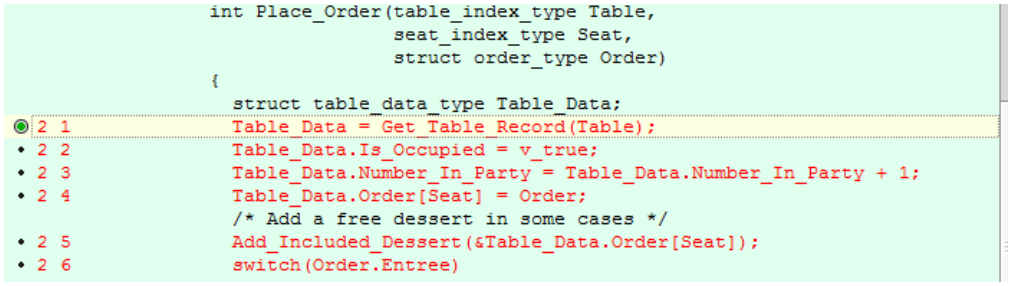
- White-Box System Testing
- Automated Fault Injection
- System Level Debugging
- Qualifiable for safety related development

Product Description

VectorCAST/Probe provides a simple way to dynamically instrument a complete application with blocks of code to enable white-box testing, inject faults, and debug hard to repeat race conditions. VectorCAST/Probe is integrated with the full family of VectorCAST tools, so that probes can be inserted during Unit, API, or System Testing, using VectorCAST/C++, or VectorCAST/QA. The probe insertions are controlled by the same technology that controls our code coverage instrumentation, which ensures that the probes function correctly regardless of the compiler, target processor, or run-time environment. There are a variety of uses for this probe technology, the following sections describes two common use-cases.

How it Works

The user simply clicks the line of code where they want to add a probe, and enters a snippet of C code. VectorCAST/Probe takes care of the compilation of the probe, the insertion into the source code, and the build of the instrumented application. Additionally, VectorCAST/Probe provides all of the string conversion, and IO functions necessary for the capture, conversion to ASCII, and output of the captured data, regardless of the run-time environment or target. Any data that is output by the probe is captured in a special section of the VectorCAST test report.



Because the probe insertion is built on the same technology as the VectorCAST/QA code instrumentation, the application of the probes is deterministic, and qualifiable for safety related development such as DO-178, IEC 61508 or ISO 26262.

VectorCAST/Probe Use Cases

Testing the Error Path

A Probe Point can force variables to be set to unexpected values or to explicitly induce a software fault such as divide by zero, stack overflow or clock rollover. By explicitly inducing the fault, the error logic will executed, and the recovery procedure can be validated.

Capturing Debug Data at the Time of Failure

When a test case fails, the tester can often see the cause of the error. A tester can validate a potential fix by creating a probe point and rerunning the test.

Separating Test Code from Production Code

By adding test code directly into the source, you run the risk of leaving test code in the end product. A better way is to use VectorCAST/Probe to maintain the test code separately from your production code.