



Automated Unit Testing of Embedded ARM Applications

Author:

Wolfgang Schmitt, Manager, Hitex

Synopsis:

The performance of today's ARM microcontrollers allows for more features and functionality to be implemented in software. Unlike a hardware implementation, software appears easier to change to accommodate changing software requirements and specifications. On the other hand, the debug capabilities available for the typical developer often only comprises run control, e.g. breakpoints and variables watch, but lacks tracing or code coverage analysis of the application. This deficiency is because the built-in debug features of the ARM microcontrollers allow for easy use of inexpensive, but feature-limited, debugging tools. This holds even more true if the Embedded Trace Macrocell (ETM) is not implemented in the ARM derivative in question.

Testing of applications at unit level allows finding bugs early in the development process and increases the confidence level for system test, because the system is now based on well-tested units. The usage of appropriate test tools enables the automation of unit testing and is the base for regression testing, thus optimizing the development process and providing the base for future product feature enhancements.

Unit Tests Remove Test Complexity

To cope with limited debug functionality, tests that reduce the (test) complexity of the whole application by dividing it into manageable (testable) pieces can be useful. This approach is called unit (or module) test. When considering an embedded application written in the C programming language, a unit is a C-level function. Unit testing isolates a single function from the set of all functions of the application and rigorously tests this function (or unit). Rigorous means that the test cases are specially made for the unit in question and also comprise input data, that may be unexpected by the unit under test. Isolation from the rest of the application can be achieved by directly calling the unit under test and replacing the calls to other functions by stub functions. If a test fails, the cause of the failure can easily be identified, because it must stem from the unit under test and not from a function further down in the calling hierarchy.

Regression Testing is a Necessity

Regression testing is the repetition of already passed tests after bug fixes or improvements in the software took place. Regression testing proves that a change in the software did not result in any unexpected behavior. Regression testing is key to software quality.

Obviously, regression testing in practice requires the automation of the tests, because the effort to repeat the tests manually is too high. Even for non-repetitive unit tests, the proper tool support will save you lots of time, but tool support is indispensable for the repetition of the unit tests. Tessy is a tool that automates most of the tasks of unit testing and also enables you to run regression tests without user interaction. Tessy analyzes your C source code and determines the interface of the units

you want to test. The interface consists of the input variables to your unit (i.e. read by the unit) together with the output variables of the unit (i.e. written by the unit). The structure of a test case follows from the structure of the interface. Tessy lets you enter test data by means of a user-friendly, graphical user interface (no script language to learn). Then Tessy compiles and links the original source code of your application together with parts generated by Tessy, using your cross compiler for ARM microcontrollers, e.g. the ADS.

Embedded Software Should be Tested on the Hardware

For embedded software it is essential that the unchanged source code with all the non-ANSI keywords is used for testing. However, it is also essential that the test executes on the actual hardware. For this purpose, Tessy is tightly integrated with Tanto for ARM, the debug platform from Hitex for ARM microcontrollers. Tanto is operated by HiTOP, the debugger for ARM from Hitex. Tessy instructs HiTOP to load the ARM binary via Tanto into the hardware and to execute the tests using the ARM microcontroller. During the unit test, the function under test can access the hardware as usual. Tessy automatically provides the test input data and evaluates the results against the expected ones. From this, Tessy creates comprehensive test documentations in several formats, e.g. Word or HTML.

After all tests are successfully conducted, it is useful to determine the test coverage. Tessy can determine the branch coverage (C1 coverage) by a click of the mouse. Branch coverage reveals if all branches present in a unit under test were taken at least once during the tests. Standards for safety-critical systems often require complete documentation of the reached test coverage.