hitex
EMBEDDED TOOLS & SOLUTIONS

TESSY Automated dynamic module/unit and integration testing of embedded applications

CTE Classification Tree Editor for test case specifications

Automated module/unit testing and debugging at its best

TESSY – The Invaluable Test Tool

TESSY performs automated dynamic module/unit and integration testing of embedded software and determines the code coverage along the way. This kind of test is required for certifications according to standards such as DO-178, IEC 61508, ISO 13849, IEC 62279, or ISO 26262.

Integrated with TESSY is the Classification Tree Editor (CTE), a tool to specify test cases systematically according to the Classification Tree Method (CTM).

TESSY is based on Eclipse Rich Client Platform (RCP) thus featuring the well-known Eclipse user interface with views and perspectives.

TESSY helps developing safety critical applications and devices

Key Features

- Automated test execution
- Test report generation
- Code coverage without extra effort
- Regression and integration testing
- Traceability of requirements to test cases
- Qualified to be used in safety-related software development
- Testing on host or actual hardware
- Supports C and C++
- Global, permanent license
- Fastens development – pays off quickly

www.hitex.com/tessy

TESSY automatically executes the tests, evaluates the test results, and generates the test reports.

TESSY can eliminate manual testing and therefore saves the embedded development engineer a tremendous amount of time. A faster development process ensures the tool recovers its costs quickly, and what's more, produces better quality software and documented tests.

TESSY is successfully used in large projects with dozens of users in multiple locations across the world. TESSY is used extensively in automotive, aerospace, avionics, railway, medical, military, and industrial applications. Ask for a testimonial!

TESSY can be operated without user intervention what is useful for regression testing and continous integration.

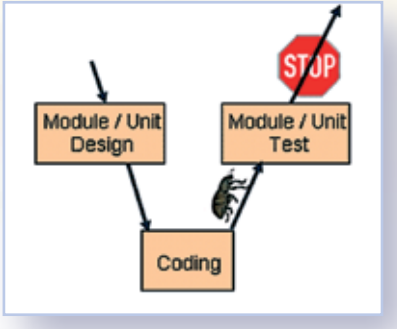
Systematic, rigorous and isolated testing

What Is Module/Unit Testing?

During unit testing of C programs, a single C-level function is tested rigorously and in isolation from the rest of the application. Often unit testing is also called module testing.

Rigorous means that testing is tightly focused on the unit in question; the test cases are specific for the unit, demanding, comprehensive, using boundary values and values that are likely to reveal a problem in the unit under test.

Isolated means that the test result does not depend on the behavior of the other units in the application. It can be achieved by directly calling the unit under test and replacing calls to other units by stub functions.



Unit testing eliminates defects early on and prevents them from showing up in later stages of the development process.

What Are The Benefits Of Module/Unit Testing?

Reduces Complexity of Test Case Specification
Instead of trying to create test cases that test the whole set of interacting units, the test cases for unit testing are specific to the unit under test (divide-and-conquer). Test cases can easily comprise of input data that tests the error handling of the unit under test, something which may be hard to achieve during system testing.

Easy Defect Isolation
If the unit under test is tested in isolation from the other units, detecting the cause of a failed test case is easy. The defect must be in the unit under test.

Finds Defects Early

Unit testing can be conducted as soon as the unit to be tested compiles successfully. Therefore defects inside the unit can be detected very soon after their creation.

Saves Money
It is generally accepted that defects detected late in a project are more expensive to correct than defects that are detected early. Hence unit testing saves money.

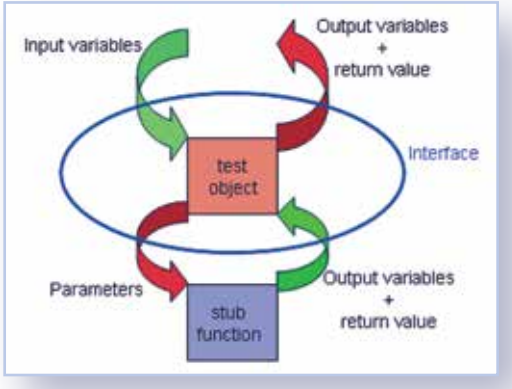
Gives Confidence
Unit testing gives confidence. After the unit testing, the application will be made up of fully tested units. A test for the whole application will then be more likely to pass.

A tour around the testing workflow

1. Unleash It For A Test Run

TESSY starts off by analyzing the source module and then lets the user select the function to be tested. It then identifies the interface of the test object, such as global variables, parameters, and functions called by the test object.

TESSY determines whether a variable of the interface is input, output, or both. TESSY can be directed to allocate memory to be used as target for pointers in the interface.



The interface separates the test object from the rest of the application

2. State Your Case

TESSY provides several ways to specify values for the test cases:

- Interactive test data input is accomplished in the tabular Test Data view
- Test data import in various file formats (including Excel)
- Test cases specified systematically by use of the Classification Tree Method can be imported from the Classification Tree Editor (CTE)
- TESSY can generate test cases from value ranges
- TESSY can generate random test data

Test data can be entered interactively

A test case comprises values for input variables, the expected results, and how to compare the actual results with the expected ones to determine if a test has passed or failed. Deviations of the expected result may be allowed.

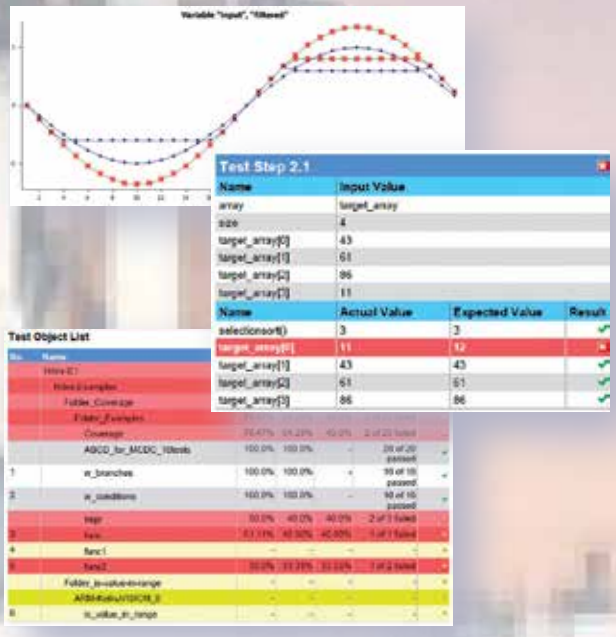
3. Meet Your Test Driver

TESSY then generates source code for the test driver, which calls the function under test. If this function calls another function, TESSY is able to create a stub function to replace the called function. This is necessary for unit testing in its strictest sense and useful if the called function is not implemented yet.

TESSY provides two types of stub functions:

- One type allows the user to specify expected values for the input variables of the stub function which are compared with the actual values by TESSY. Furthermore this stub function type allows you to specify the inputs from the stub function to the function under test, e.g. the return value.
- The other type of stub function allows the user to provide source code for the body of the stub.

TESSY can also check the order of the stub function calls.



Test reports are generated automatically

4. Go TESSY Go!

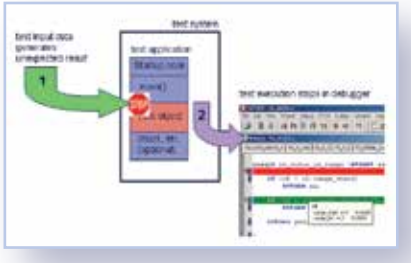
Using a cross compiler, TESSY compiles and links the driver source code, the function under test and any stub functions, and then downloads the resulting executable to the test system. This might be an in-circuit emulator in stand-alone mode or one connected to a target system, or a JTAG / BDM / OCDS debug system. This might also be a simulation of the target microcontroller running on the host PC. Testing can also be performed on the host PC using the native GNU compiler.

TESSY executes each test case and then determines, if it has passed or failed. A comprehensive test report can be created in pdf, word, html and xml format.

Why TESSY eases testing

Dream Debugging

If a test case fails, an easy and efficient debugging is in place. TESSY is able to re-execute a test case and direct the debugger in use to stop test execution at the beginning of the function under test. The debugger's features now can be used to reveal the culprit. After the source code is changed to fix the bug, the test case in question (and all others) can easily be re-run to verify that the correction operates successfully.



TESSY eases debugging of a failed test case

Re-Use Test Data And Save Time

If any interface element of a tested function has been changed in the course of the development process, TESSY allows the user to re-use test data from the old interface, which considerably aids the regression testing process.

Regression Testing – Did Your Modifications Cause Errors?

Regression testing can reveal if new errors have been introduced during further development of the application, such as bug fixes in other sections, rewriting of the tested function, switching to a new compiler version or porting the software to another microcontroller architecture. TESSY's easy-to-use regression testing ability is an extremely helpful method of checking modified software and thus ensuring software quality.

Non-interactive Testing – Lets You Go Home

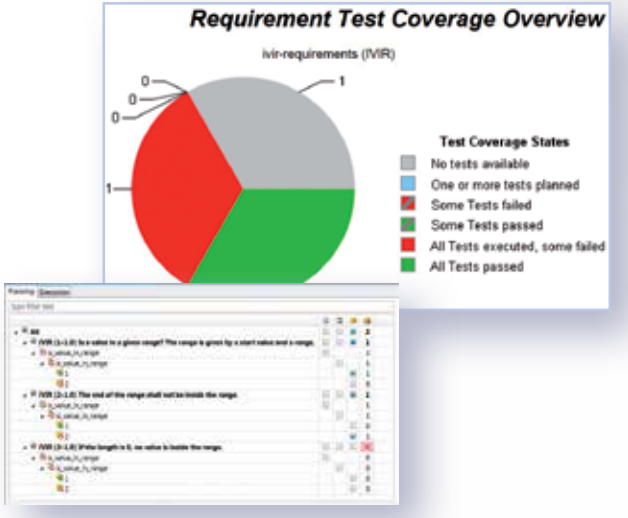
TESSY features a command language allowing the user to run a selected set of test cases without any user intervention. So an extensive regression test can be run overnight and the results can be analyzed the next day.

ASAP2 Files Recognized

TESSY recognizes ASAP2 files, which enables the user to use physical values (e.g. the temperature in degrees Celsius) instead of the internal representation.

Trace Your Requirements

Traceability of requirements to test cases checks if all requirements have a test case associated with them and also lets you find out which test cases may need adaptation if a requirement has changed. Requirements can be imported / exported and created in TESSY.



Do all requirements have a test case linked to them?

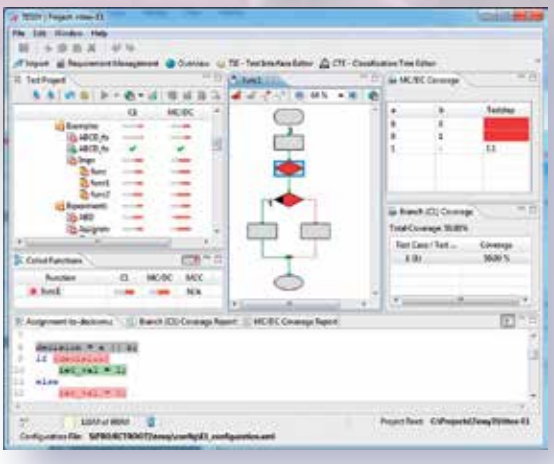
Software quality needs TESSY

Code Coverage – Ensure Everthing's Tested

Without additional effort, TESSY can determine the following code coverage measures:

- Entry Point Coverage
- Statement Coverage
- Branch Coverage
- Decision Coverage
- Modified Condition / Decision Coverage (MC/DC)
- Multiple Condition Coverage
- Function Coverage (for integration testing)

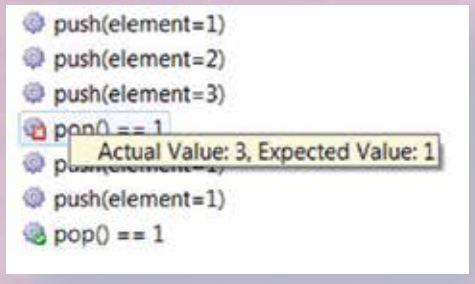
TESSY's coverage viewer shows the results. The user can interactively display the source code related to a certain branch or decision of the software. This reveals with a click of the mouse which branch of the software was not executed during the tests, or which path was executed by a certain test case.



Investigating the coverage results is interactive

Integration Testing – Check The Interaction

TESSY can test the interaction of cooperating units even if the units do not call each other, e.g. push() and pop() of a stack. Several units are composed to form a bigger unit, usually called component. A test case calls the units in a certain order and sets variables of the component. Resulting calls to other components and variable values can be checked by TESSY. Integration test cases can also include simulated time (temporal component testing).



An integration test case

Supported Microcontrollers

TESSY is adapted to currently more than 150 combinations of microcontroller / cross compiler / debugger. This ensures that TESSY is able to handle non-ANSI-C microcontroller-specific code of some cross compilers. Since TESSY controls the debuggers, TESSY can execute the tests automatically.

The list of supported combinations is extended steadily. Please check www.hitex.com/tessy.

TESSY runs on MS Windows. Both TESSY and CTE originate from the former software technology laboratory of Daimler AG in Berlin, Germany.

Visualise your testing ideas!

hitex
EMBEDDED TOOLS & SOLUTIONS

CTE And The Classification Tree Method

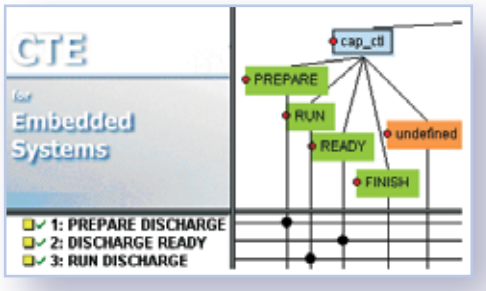
The Classification Tree Method supports a developer confronted with issues such as:

- Finding the ”right” test cases
- Minimizing a set of test cases while assuring that none are missing
- Estimating the amount of testing required
- Defining criteria needed to conclude testing without risking integrity of the test process

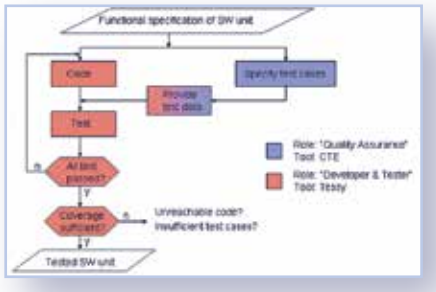
The Classification Tree Method transforms requirements systematically into a set of error-sensitive, low-redundancy test cases. This method classifies test-relevant aspects using the equivalence partitioning method and leads to test case specifications that can comprise boundary values.

TESSY is qualified to be used in safety-related software development according to IEC 61508 and ISO 26262

The Classification Tree Method is intuitive and easy to learn. It requires and encourages the developer to employ their creativity. Because thinking about the problem specification is at the very beginning, the Classification Tree Method also reveals inconsistencies or omissions in the problem specification.



Excerpt of a test case specification in the CTE



Test case specification and test execution can be separated using TESSY and CTE

The Classification Tree Editor (CTE) is a graphical tool supporting the Classification Tree Method. It is integrated in TESSY.

Although CTE is included in TESSY, its use is not only limited to embedded systems or module/unit testing.

Visit us on the internet! www.hitex.com

Main Office Germany
Greschbachstraße 12
D-76229 Karlsruhe
Tel. +49-721-9628-0
Fax +49-721-9628-149
E-mail sales@hitex.de

Hitex UK
Millburn Hill Road
University of Warwick
Science Park
Coventry CV47HS
Tel +44 (0)24 7669 2066
Fax +44 (0)24 7669 2131
E-mail: sales@hitex.co.uk

TESSY3.indd Jan 2016-010