hitex

DEVELOPMENT TOOLS

Tessy

**Automated dynamic module/unit testing
for embedded applications**

CTE

**Classification Tree Editor
for test case specifications**

Embedding Software Quality

Automated module/unit testing and debugging at its best

Tessy – The Invaluable Test Tool

Tessy performs automated dynamic module/unit 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-178B or IEC 61508.

The Classification Tree Editor (CTE) provides one method of systematically specifying and generating test cases. It makes use of the Classification Tree Method to do this, and is a unique feature to Tessy.

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
> Essential to get certifications
> For geographically distributed projects
> Testing on host or actual hardware
> Supports C and C++
> Fastens development – pays off quickly

www.hitex.com/perm/tessy.html

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.

Tests can be executed on the development host or on the actual target hardware.

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!

Regression testing is a key feature in achieving software quality and Tessy is the perfect tool for this task.

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 the test cases are specially made for the unit in question and that they comprise of input data that may be unexpected by 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.
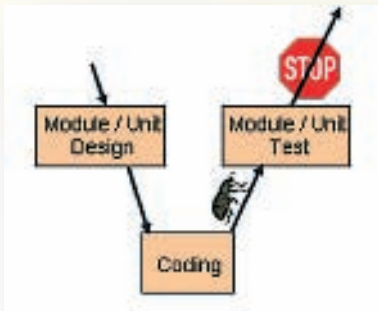
Unit testing eliminates errors early on and prevents them from showing up in later stages of the development process.

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 is unexpected by the unit under test, something which is hard to achieve during system testing.

Easy Fault 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 fault must be related to the unit under test, and not to a unit further down the calling hierarchy.

Finds Errors Early
Unit testing can be conducted as soon as the unit to be tested compiles successfully. Therefore errors inside the unit can be detected very early.

Saves Money
It is generally accepted that errors detected late in a project are more expensive to correct than errors 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 single, 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 is input, output, or both. The findings are displayed in the Test Interface Editor (TIE). There Tessy can be directed to allocate memory to be used as target for pointers in the interface.
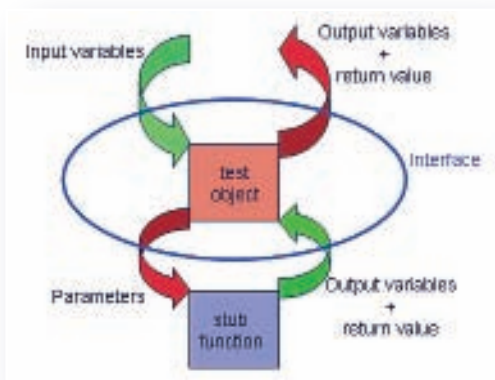

The interface separates the test object from the rest of the application

2. State Your Case

Tessy provides several ways to specify values for the test cases.

> Interactive test data input is accomplished by the wizard-like Test Data Editor (TDE).

> Tessy automatically combines user-supplied values to test cases. Test cases to cover ranges of values can be generated easily.

> Test data can be imported from files (Excel workbooks or plain text files). This allows calculating test data in Excel or to import recorded test data.

> Test cases generated systematically by use of the Classification Tree Method can be imported from the Classification Tree Editor (CTE). Those test cases can comprise test data.

> Tessy can generate random data.
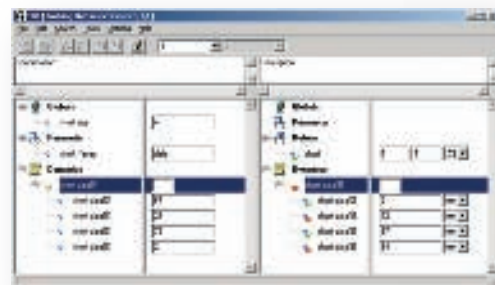
Test data can be entered interactively in the Test Data Editor (TDE)

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 now 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.
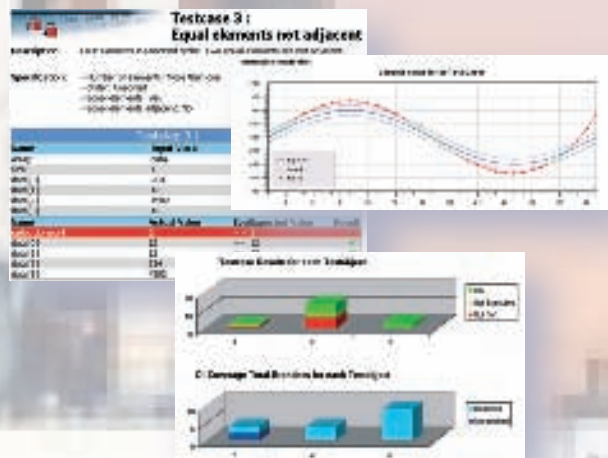
*Test reports are generated automatically*

4. Go Tessy Go!

Using a suitable 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 test report will then be created in configurable levels of detail and in various formats.

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 re-run to verify that the correction operates successfully.
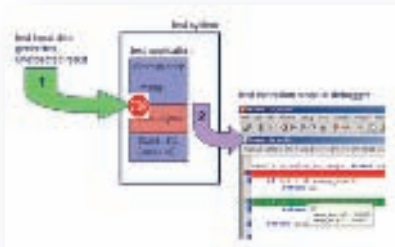
Tessy eases debugging of a failed test case

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.

Old interface elements can be assigned to new ones using the Interface Data Assign editor (IDA)

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.

Batch Testing – Lets You Go Home

Tessy allows 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.

Integration Testing – Check The Interfaces

Tessy can be used to do some integration testing. This is done by using the actual units instead of stub functions, normally in a bottom-up approach to integration testing.

Environment Editor –
Manage Your Configurations

Configurations of Tessy for a certain project can be created by the use of Tessy's Environment Editor (TEE). This enables many users to use the same configuration easily, what is useful in bigger projects.

Software quality needs Tessy
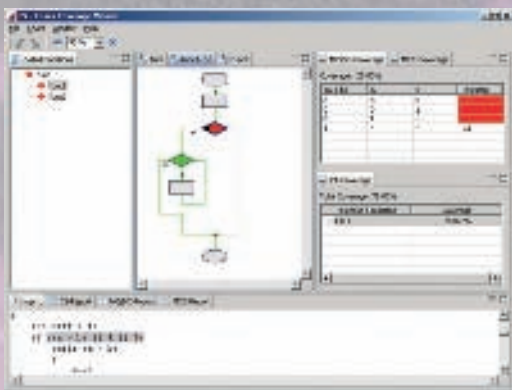
Investigating the coverage results is interactive

Code Coverage – Ensure Everthing's Tested

Without additional effort, Tessy can determine the following code coverage measures during the test:

> Branch / Decision Coverage (C1)

> Modified Condition / Decision Coverage (MC/DC)

> Multiple Condition Coverage (MCC).

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 test case did execute a certain branch of the software.

Also the truth table for conditions of a decision is displayed. This reveals if any additional test cases are needed, and if yes, how they should be made up. All coverage test results are also available as printable reports.

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 integer representation (used by the microcontroller) of that physical value. Additional information from the ASAP2 file (e.g. unit description, minimum and maximum values) may be displayed within the Tessy tools and reports.
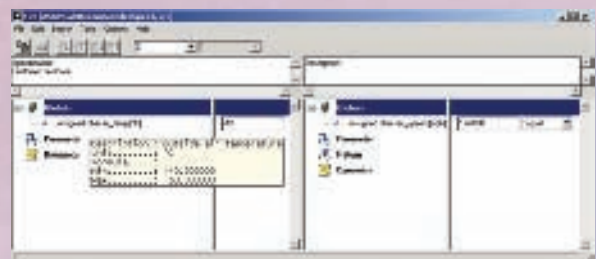
ASAP support allows using physical test values

Supported Microcontrollers

Tessy is adapted to currently more than 130 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 is adapted to various debuggers, Tessy can execute the test automatically.

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

Tessy runs on WindowsNT /2000 / XP and Vista. Both Tessy and CTE originate from the former software technology laboratory of Daimler AG in Berlin, Germany.

Design tests for confidence

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 a problem specification 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.

Excerpt of a test case specification

This approach 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.

The Classification Tree Editor (CTE) is a graphical tool that supports the Classification Tree Method. In the upper window, the classification tree is drawn. In the lower window, a line in the combination table specifies a test case. You can annotate information such as descriptions, references to requirements, expected results to the tree. Bigger trees may be split into sub-trees. The editor is able to export test case specifications to Tessy and to files.
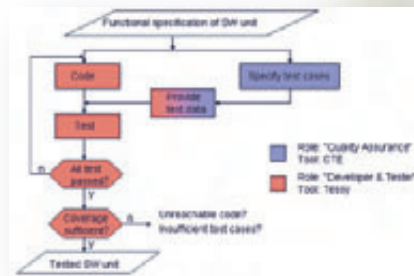
Test case specification and test execution can be separated using Tessy and CTE

If the CTE is used in conjunction with Tessy, you may assign values to classes. An assigned value is exported to Tessy with the test case specification. So test data is assigned to test cases very efficiently and comfortable. This speeds up testing a lot.

Although CTE is included in Tessy, its use is not only limited to embedded systems and it is therefore also available as a separate product.

Embedding Software Quality