The ROTTest Unit Test Tool

The ROTTest (Right On Target) tool is a unit test tool for embedded C applications. It is developed for AVR32 stand alone applications using AVR32 Studio and the AVR32 GNU Toolchain. The tool is easily ported to other environments. The tests are executed on the target platform.

The idea is to place all unit tests in a specific section in memory. The *doTests()* function executes all the tests it finds in the *rot_test_section* in the order of appearance.

ROTTest Manual

1. In the linker script the section *rot_tests* must be defined. Also the variables *rot_tests_section_start* and *rot_tests_section_end* must be defined in the linker script. Ex:

```
SECTIONS
{…
  .rot_tests     :
  {
    rot_tests_section_start = .;
    KEEP (*(.rot_tests))
    rot_tests_section_end = .;
  } >FLASH AT>FLASH
…
}
```

2. In file where the unit tests are constructed the header file *rot_test.h* must be included. Also the *setUp()* and *tearDown()* functions must be defined in this file and declared as *static*.

3. To report the status of each test a callback function must be registered. This is done with:

   *registerReportFunction(void (*callbackFunction) (char *)))*

4. A test suite must be declared as:

```
ROTTestSuite(nameOfTheTestSuite,setUp,tearDown) {
…
}
```

   where *setUp* and *tearDown* must be defined as above. All tests defined in the same file as the test suite will belong to this particular test suite.

5. A test is declared as:

```
ROTTest(nameOfTheTest) {
…
```

}

6. To verify expressions the following functionality is provided:

```
ROTAssert(expr)
ROTAssertEquals(expr1,expr2)
ROTFail(expr)
```

   These functions use the report callback function registered, as in 2. above, to report the result.

7. The *doTests()* function is the driver of the test suite.

ROTTest example

In this example a callback function that writes to the serial port is used.

The test file:

```
#include "rot_test.h"
#include "stdio.h"
#include "command_handler.h" // contains the putString(char *) function


static void setUp(void){
	;
}

static void tearDown(void){
	;
}

ROTTestSuite(testSuite1,setUp,tearDown);

ROTTest(test1) {
	ROTAssert(1);
}

ROTTest(test2) {
	int toBeExecuted = 0;
	int notToBeExecuted = 1;
	ROTAssert(toBeExecuted);
	ROTAssert(notToBeExecuted);
}

ROTTest(test3) {
	int x = 1,y = 1;
	ROTAssertEquals(x,y);
}

ROTTest(test4) {
	int x = 0, y = 1;
	ROTAssertEquals(x,y);
}

ROTTest(test5){
	ROTFail("Should fail");
}
```

Snippet from the *main(void)* function where the *doTest()* is called:

```
#ifdef TEST
	registerReportFunction(putString);
	doTests();
#endif
```

The test report at the serial port:

```
Start Tests
	Start TestSuite: Name: testSuite1
		Start Test: Name: test1
			Assert Passed
		Completed Test: Name: test1 Asserts Passed: 1 Asserts Failed: 0
		Start Test: Name: test2
			Assert Failed: Filename: ..\src\test.c Line: 28 Expression: toBeExecuted
		Completed Test: Name: test2 Asserts Passed: 0 Asserts Failed: 1
		Start Test: Name: test3
			Assert Passed
		Completed Test: Name: test3 Asserts Passed: 1 Asserts Failed: 0
		Start Test: Name: test4
			Assert Failed: Filename: ..\src\test.c Line: 39 Expression: x == y
		Completed Test: Name: test4 Asserts Passed: 0 Asserts Failed: 1
		Start Test: Name: test5
			Assert Failed: Filename: ..\src\test.c Line: 43 Expression: "Should fail"
		Completed Test: Name: test5 Asserts Passed: 0 Asserts Failed: 1
	Completed TestSuite: Name: testSuite1 Tests Passed: 2 Tests Failed: 3 Asserts Passed:
	2 Asserts Failed: 3
Completed Tests: Passed: 2 Failed: 3
```