On Target Unit Testing

Overview

One of the main requirement for effective firmware development is moving test before construction than after construction.

Unit testing is a method by which individual units of source code are tested before or during construction. Test Driven Development advocates writing test code before application code, thus bringing all the concepts of structured testing to the average programmer (rather than being seen as a separate activity).

C is the dominant programming language for embedded systems, but most of the unit testing framework are written in Java or C++. The majority of leading framework prove problematic when looking at testing in the context of an embedded application.

Of course it is possible to test a C program using a C++ test framework, but this isn’t always suitable due to the change of programming paradigm or due to a lack of complier support.

The goal of this document is to introduce the C unit testing framework and using it to do on target unit testing using Keil and STM32 as platform.

Requirements

* Using C for unit test instead of C++.
* Not library based and simple to use.
* Embeddable – supports redirection of output over serial port.
* Works with system having small SRAM.
* Works with all commercial cross compilers.
* OS independent – not Linux centric
* Single toolchain for development and unit testing – eliminates the barrier of having to ‘context switch’ to another tool and way of thinking in order to test their code.

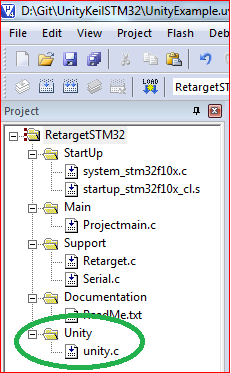
Testing Framework

Unity is an open source lightweight test harness that can be used for in-target testing of an embedded C application.

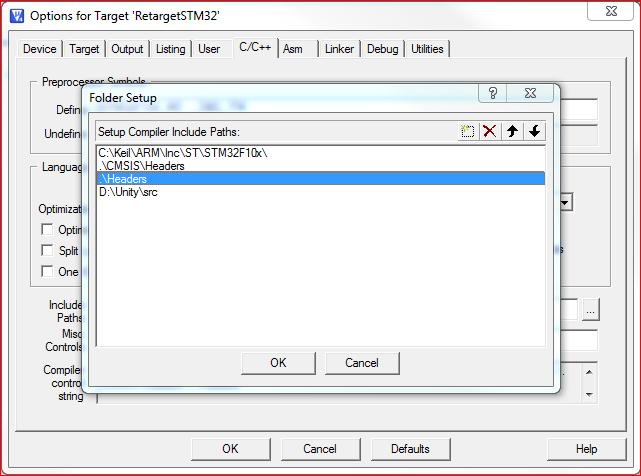
Setting up Unity in Keil uVision for STM32

Get Unity Testing Framework

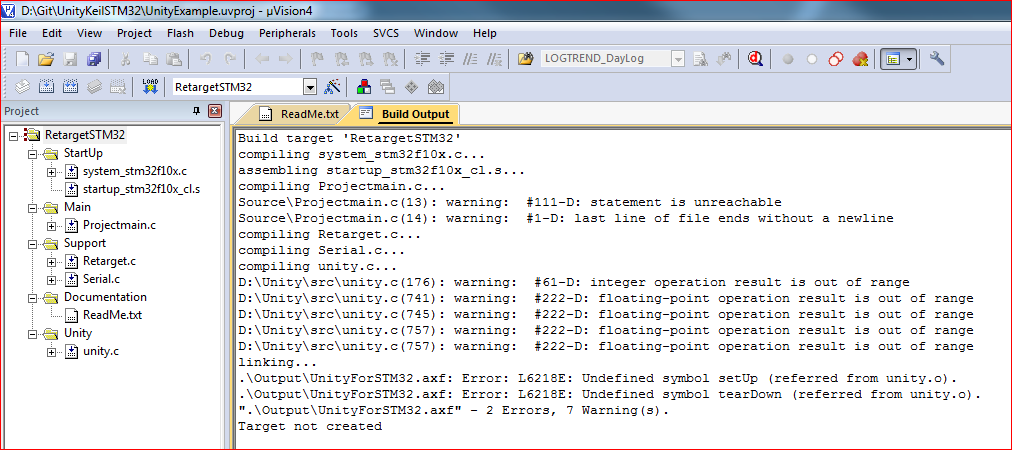
1. Clone unity from github - <https://github.com/ThrowTheSwitch/Unity>
2. Preferably have in a common place for your workspace. (C:\Unity) The simple path makes life easier.
3. Following files will be found in C:\Unity\src
   1. unity.c
   2. unity.h
   3. unity\_internals.h
4. Create a group in Keil project (used example project from <https://github.com/vigneswaranj/KeilRetargetExample> in this documentation) and add unity.c from c:\unity\src.



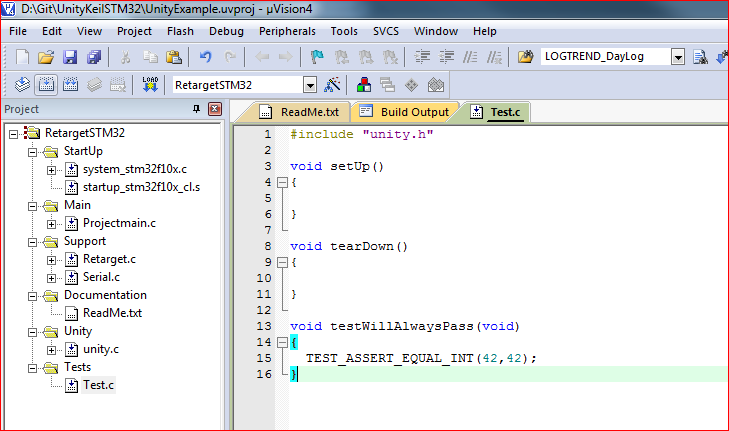
1. Include unity.h path to the Keil project options ->C/C++ -> Include paths.



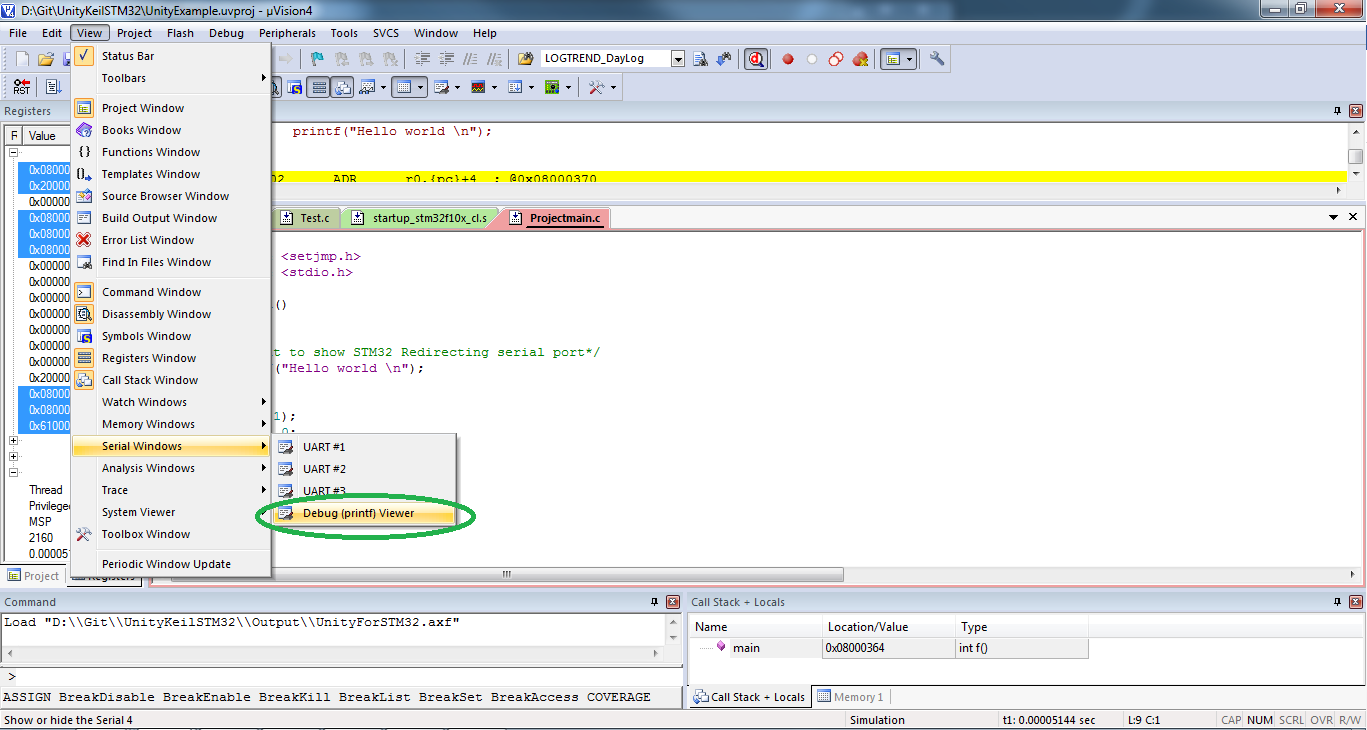
1. If the Keil project is built now it will give link errors. Ignore warnings for this example.



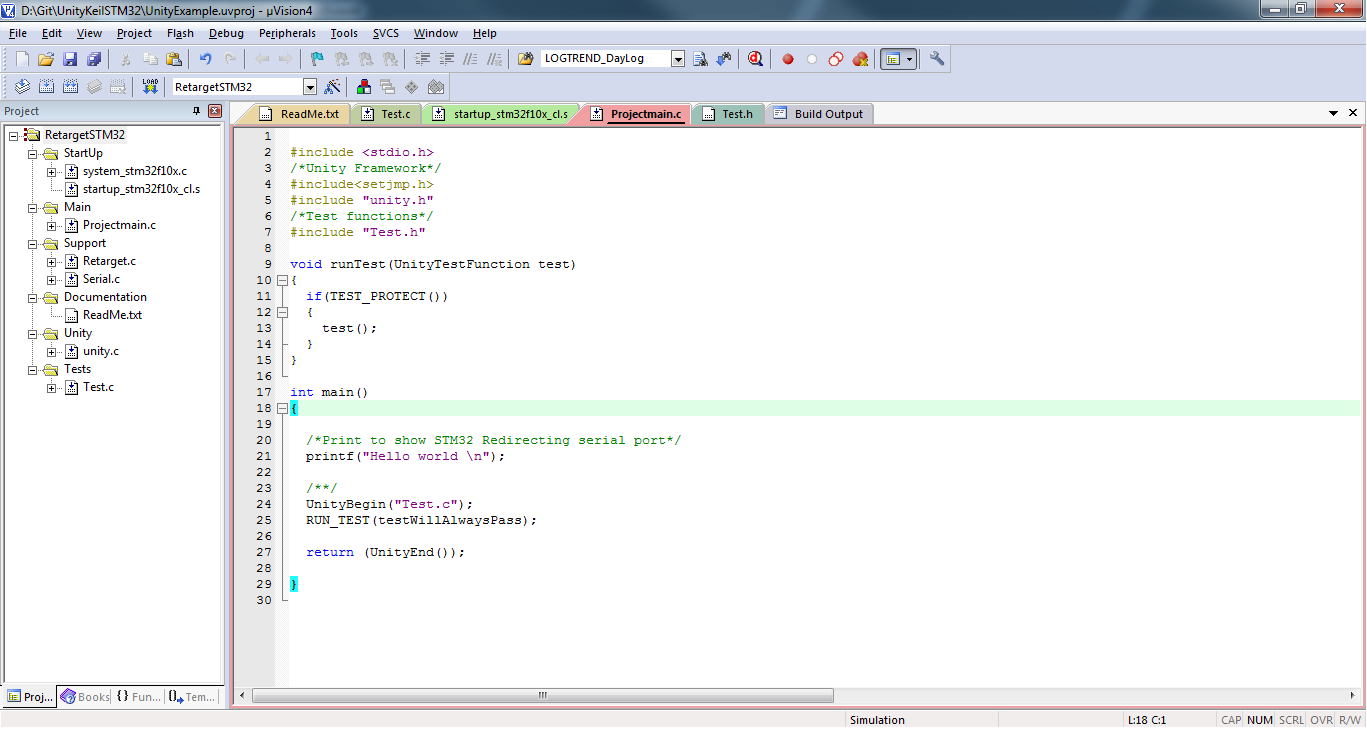
1. Now we need to build a simple test. Create test.c and add this to the test group in project as shown below.



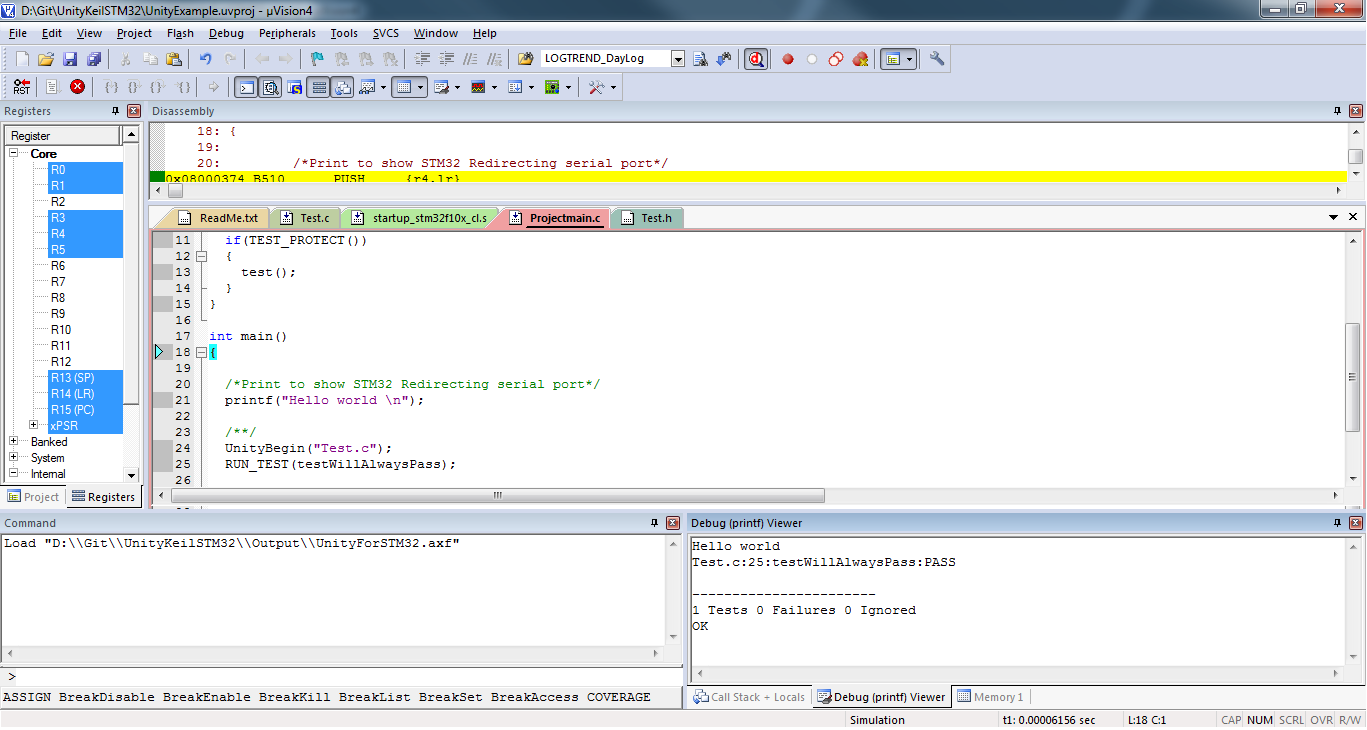
1. Note we need to create (empty) setup and teardown functions as the unity framework expects these functions (without them it will fail to link).
2. Now a simple test runner has to be created to run the test. Unity uses setjmp/longjmp to manage the tests, so we need to include setjmp.h from the C standard library.
3. By default the reporting mechanism of unity uses the function putchar from stdio.h. Since we redirected this already in our project we will get the results from Debug printf view in keil.



1. Include header test.c which has the prototype for the test function.
2. Unity requires you to define a function called runTest that takes a function pointer as a parameter. The TEST\_PROTECT macro wraps the use of setjmp/longjmp to manage test that fail or are aborted.



1. Now the project is ready to run unity test. The output from the unity test will now be displayed in the terminal I/O window.



Result

We achieved most of our requirements, the same project can be tested in the target microcontroller by adding addition hardware initialization.