Writing Managed Code Tests

# Test components

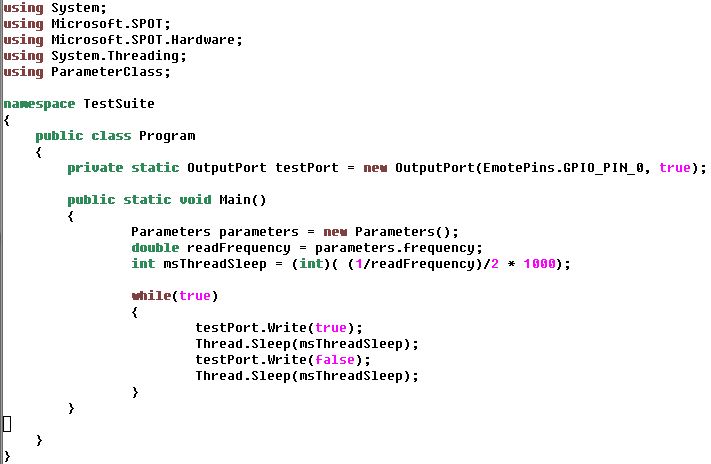
The following are the components that make up a managed code test:

* Test code
* Test description file
* Test parameters
* Test analysis script
* Test hookup file
* Test event definition file

## Test Code

The test code is C# code that will run on the TinyCLR microframework. A template C# test can be found in your TestSys directory at: “TestSys\tools\testCreate\TemplateFilesCSharp”.

Your test code can be placed in “Program.cs” starting in the function “Main()”. An example test that toggles a simple GPIO output line is shown here and in “TestSys\GPIO\Src\C#”:



## Test Description File

Every test must contain a test description file and be named, “tests.xml”. A template test description file can be found in your TestSys directory at: “TestSys\tools\testCreate\TemplateFilesCSharp”. This file informs the test tool of necessary compilation and path information in order to run your test properly.

The fields for the test description file are as follows:

* Name – name of the test
* Type – either ‘C#’ or ‘Native’
* Description – whatever you want; can be used to help filter your test list when selecting your tests to be run
* TestPath – the path to the directory of the test being written
* TestProjName – name of your project file
* TesterName – name of person queuing test (will automatically be populated by TestRig tool if left blank)
* TestLocation – name of the location queuing up the test (will automatically be populated by TestRig if left blank)
* TestMFVersion – version of Micro Framework to compile and use (currently can be ‘4.0’)
* TestGitOption – can be used to override the user selection in the TestRig; (Use local code; Use archive code; Use archive branch code)
* TestGitBranch – can be used to override the user selection in the TestRig; name of GitHub branch
* TestUsePrecompiledBinary – instead of compiling the Micro Framework source file a precompiled binary can be specified here; the precompiled \*.axf binary needs to be placed in your test directory and the full file name written here; leave blank if not needed

A sample test description file follows:



## Test Parameters

The test parameter file contains parameters that are used by your test code, the TestRig tool, and your test analysis scripts. The proper way to include your “Parameters.cs” file is shown in the Test Code example shown previously and in the template C# test.

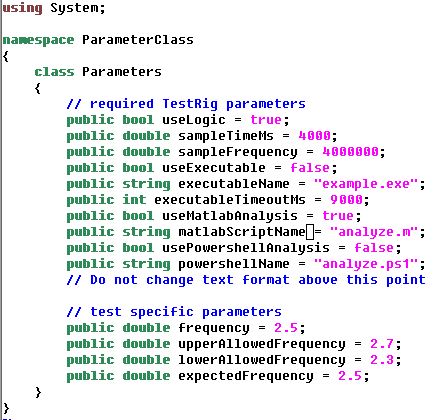
The test parameters allow the user to either use the logic analyzer code built into the TestRig or to use a custom executable to generate test data. To enable the logic analyzer code, set the parameter “useLogic” to “true” and provide appropriate values to the “sampleTimeMs” and “sampleFrequency” variables. To enable a custom executable provided by you, set the parameter “useExecutable” to “true”, provide the executable name in the parameter “executableName”, and finally provide a timeout to the “executableTimeoutMs” in case the provided executable encounters a problem and must be forced to quit (an error will be logged in the test receipt if this were to occur).

All data generated during the test is to be saved in a directory that will be temporarily created during the test: “<C# project directory>\testTemp”. Any data generated by a custom executable is to be placed within this directory which will be deleted after the test is complete.

After the logic analyzer or custom executable are finished an analysis script will be run. At the moment only Matlab scripts are used. Make sure the parameter “useMatlabAnalysis” is set to true and the name of your Matlab script is set at the “matlabScriptName” parameter.

Parameters to be used by your test script and test program are to be set under the “// test specific parameters” comment.

An example “Parameter.cs” file follows:



## Test analysis script

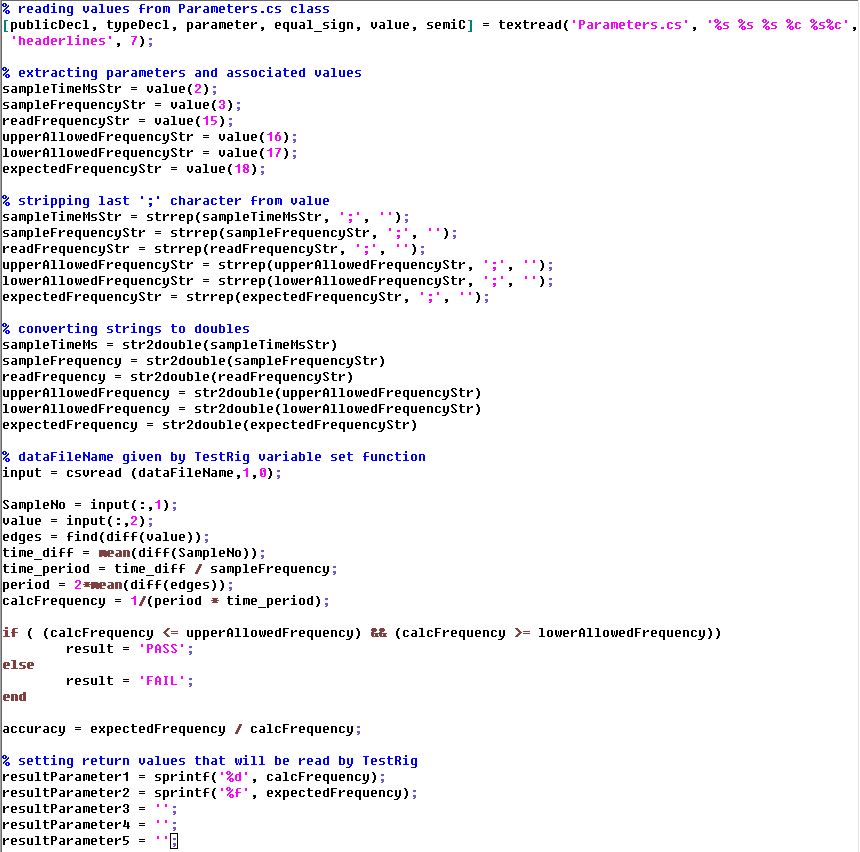
The test analysis script will analyze any collected data and provide a Pass/Fail result, an accuracy result, and some general test feedback to be saved in the test receipt.

The parameters that were passed to both the test code and the TestRig tool need to parsed by the analysis script and used to determine whether the test passed or failed. The test data must be placed within the “testTemp” directory. The logic analyzer data will be automatically stored in a file called “testTemp/testData.csv”. This filename will also automatically be given to the Matlab variable “dataFileName”.

At the conclusion of the test analysis seven variables must be set. These variables will be saved in the test receipt. The variables to be set are:

* result – a ‘string’ variable that must be set to either ‘PASS’ or ‘FAIL’
* accuracy – a ‘double’ variable that contains a test writer determined value
* resultParameter1 through resultParameter5 – five ‘string’ variables that can contain whatever data the test writer would like included in the test receipt

An example Matlab script that parses ‘double’ variable parameters and calculates the toggle frequency of one of the GPIO pins follows:



## Test hookup file

The test hookup file describes what pins will be sampled by the TestRig tool. An example of a hookup file that saves data from GPIO pin 1 follows. The file will have the same root name as the project except it will end in \*.hkp instead of \*.csproj.



## Test event definition file

The test event definition file describes what data events will be saved. An example of an event definition that will save all data line transitions on a GPIO pin follows. The event definition file will have the same root name as the project except it will end in \*.edf instead of \*.csproj.

