

Using TASKING Script Debugger

Abstract

This document describes the setup and handling of the TASKING Script Debugger which is used when selecting **TASKING VX** as target from within the TESSY Environment Editor (TEE). The TASKING Script Debugger is used for normal test runs. If you need to interactively debug your test object, refer to chapter 5.

Table of Contents

Abstract	1
1 Introduction.....	2
2 Create the Debugger Configuration File	2
3 Setting up the TEE	5
4 Debugger Launch Path.....	6
5 Interactive Debugging.....	7
5.1 Creating a test application to debug	7
5.2 Debugging the test application	8
5.2.1 Importing the test binary.....	8
5.2.2 Debugging the test binary	12
6 Troubleshooting.....	17
6.1.1 The script debugger runs into a watchdog/trap	17

1 Introduction

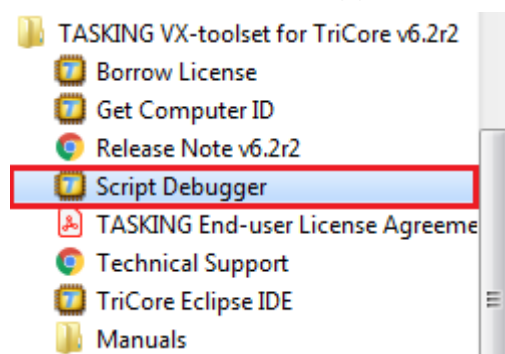
Using the TASKING Script Debugger as TESSY's target debugger requires some pre-adjustments to be performed. You have to

- set the correct paths to the compiler installation directory and to the debugger installation directory within TEE,
- find the proper startup code and linker file,
- copy the `MConfig` file from your TASKING project into the `config` directory of your TESSY project directory (see chapter 4 for further details),
- and you will need a proper debugger configuration file.

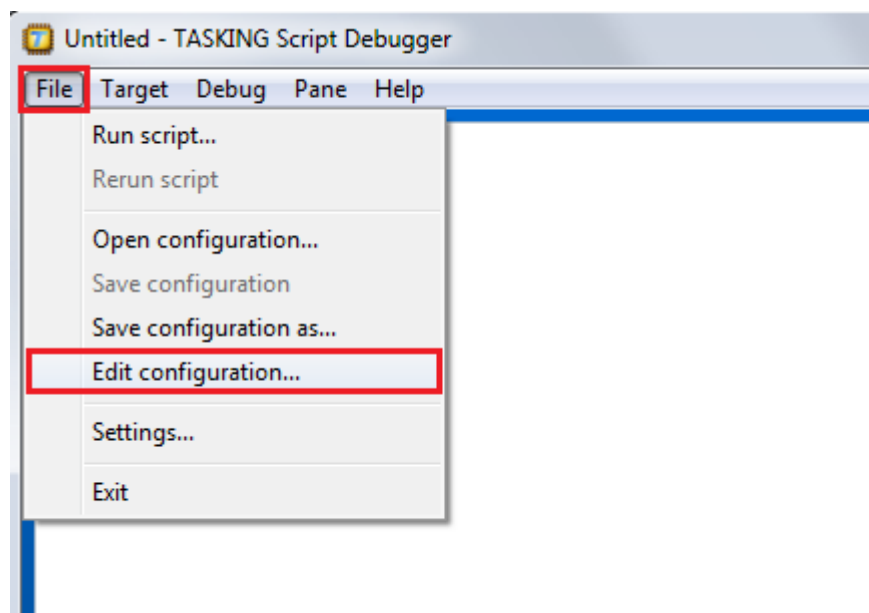
How to create the latter one is described in this document. Application note *021 Tasking Compiler* covers the compiler and linker settings.

2 Create the Debugger Configuration File

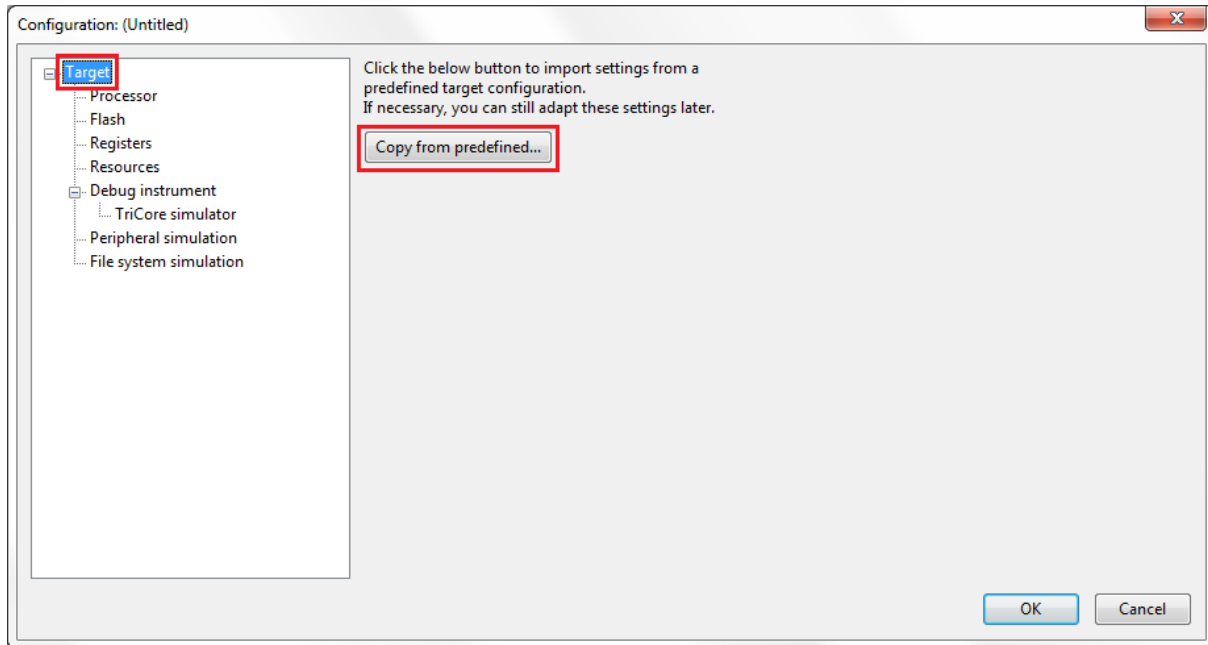
Launch the script debugger from your Windows menu.



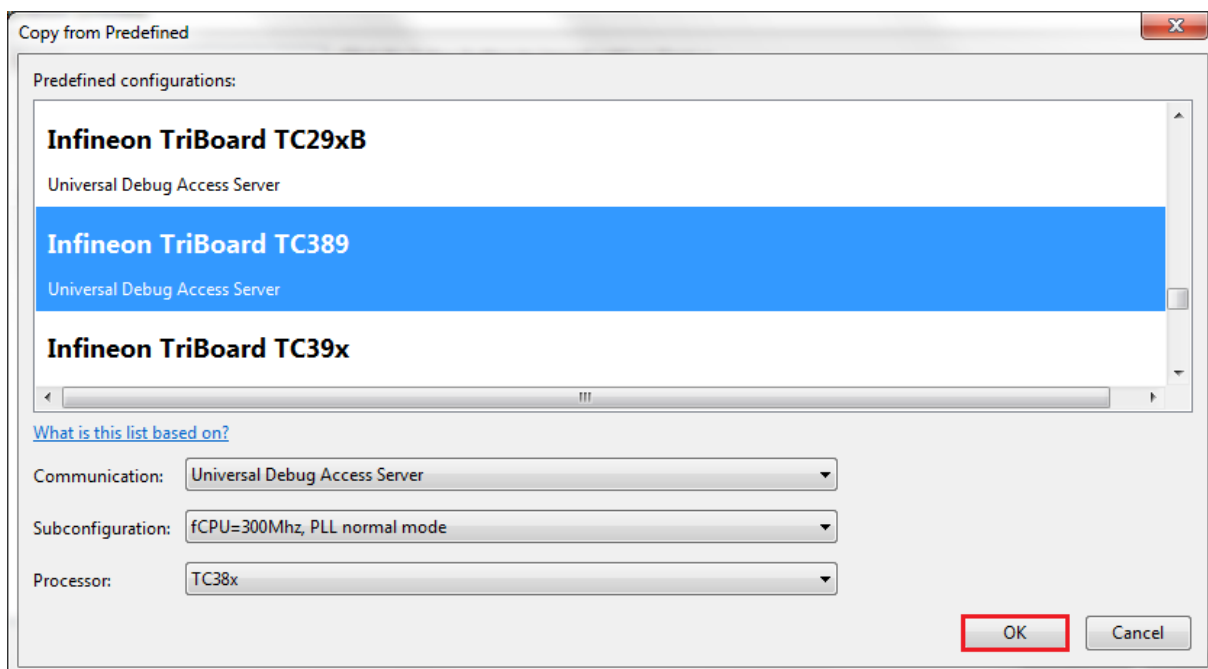
From the TASKING Script Debugger menu choose **Edit Configuration...**



From the **Configuration** dialog select **Copy from predefined...**

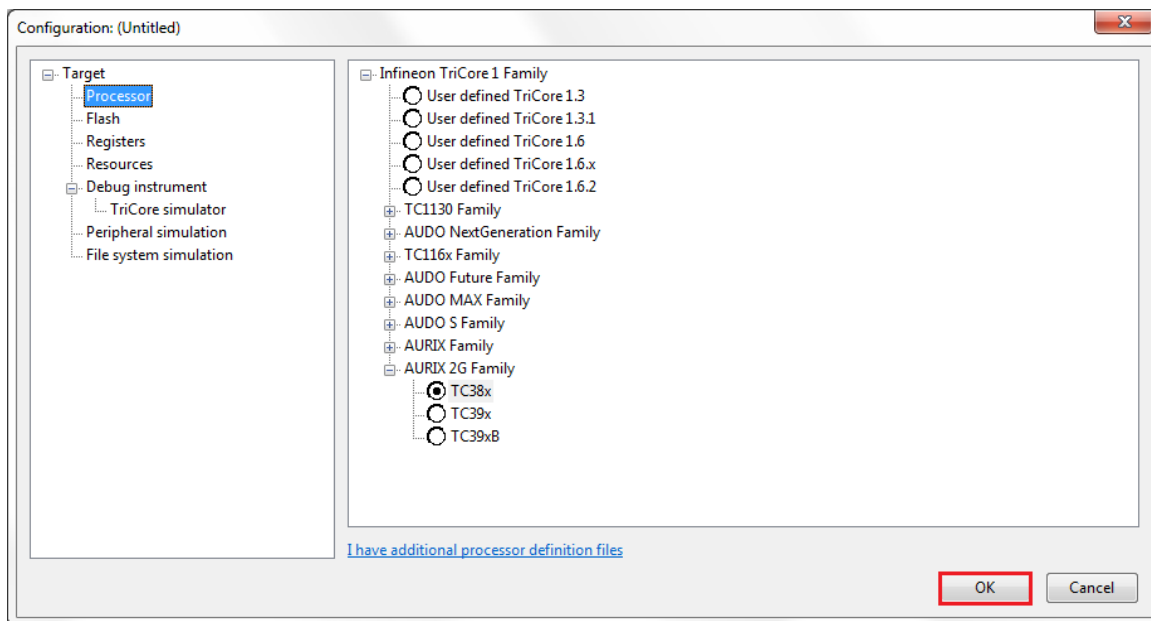


From the **Copy from Predefined** dialog choose your board. In our example we choose *Infineon TriBoard TC389*.

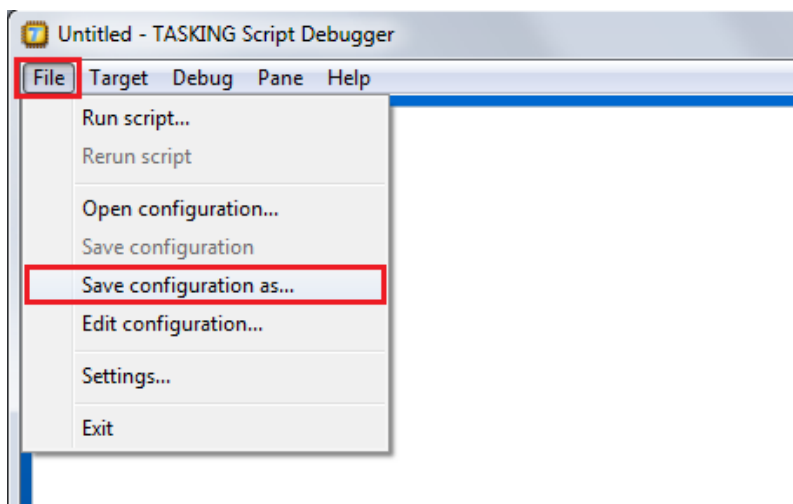


Click **OK**.

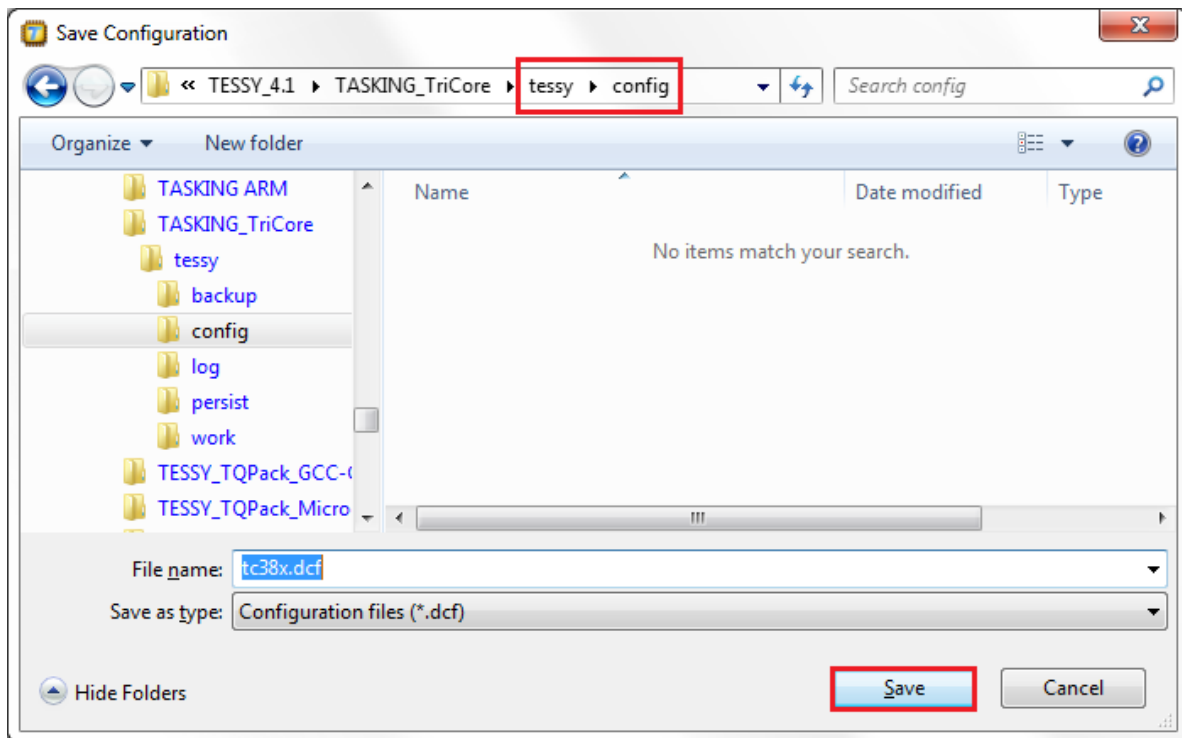
Now you may check if further adjustments are needed. Finally click OK.



From the File menu select **Save configuration as...**

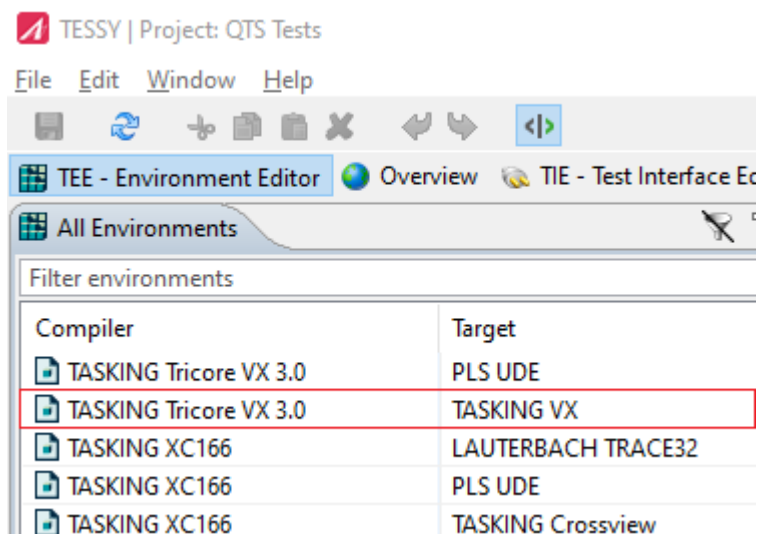


Save the file into your TESSY project's *config* directory.

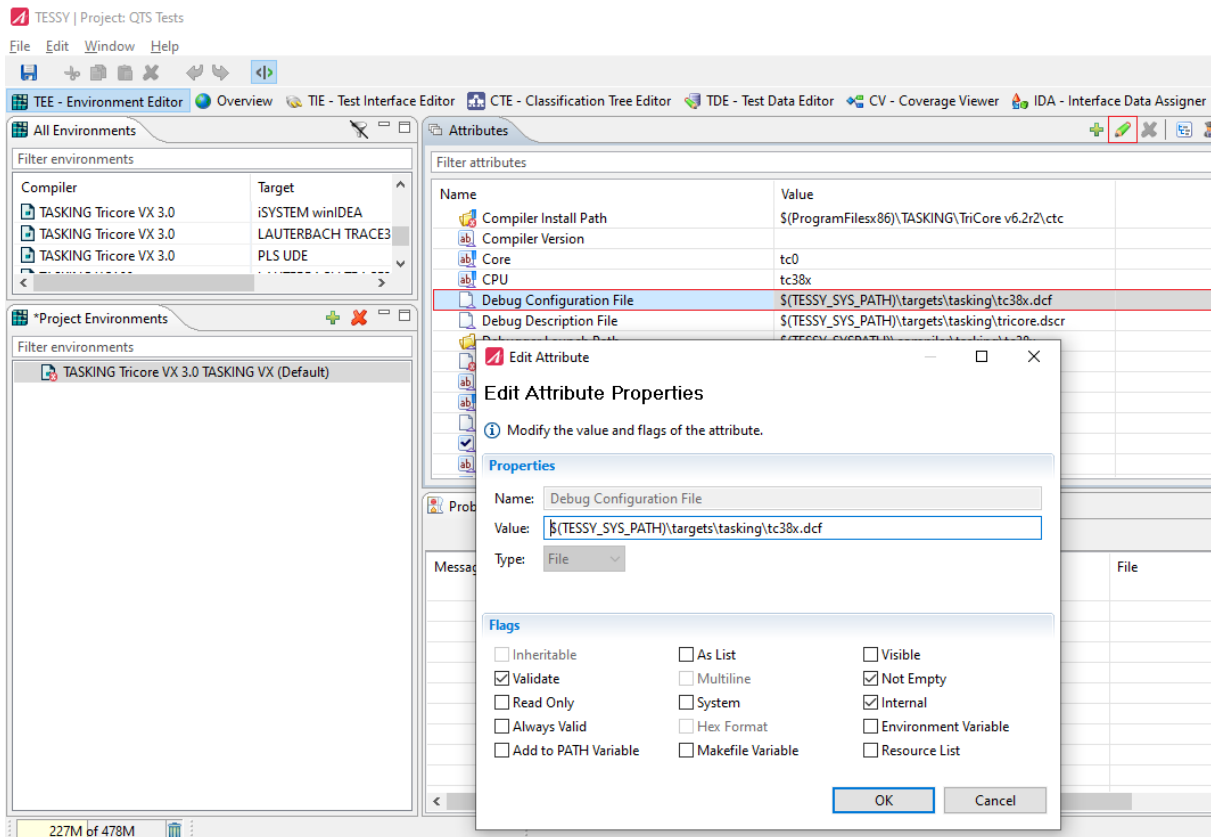


3 Setting up the TEE




Open TESSY, if you have not done by now, and launch the TEE. From TEE's **All Environments** tab select **TASKING VX**.



Scroll down to attribute **Debug Configuration File**. Edit the attribute value and select the debug configuration file you just saved.



The resulting attribute values should look like this.

 Debug Configuration File	<code>\$(Debugger Launch Path)\tc38x.dcf</code>
 Debugger Path	<code>\$(Target Install Path)\bin\dbgtc.exe</code>
 Target Install Path	<code>\$(ProgramFilesx86)\TASKING\TriCore v6.2r2</code>

4 Debugger Launch Path

The debugger launch path is **not** the installation path of the TASKING debugger, but instead the directory from where the TASKING debugger will be launched by TESSY. This directory has to contain the `MConfig` file, which is found in your TASKING development project. The TEE attribute **Debugger Launch Path** points to your TESSY's project configuration directory, i.e. `$(PROJECTROOT)\tessy\config`. It is strongly recommended to keep it that way, because it enhances the portability of the TESSY project. So, the best way is to copy the `MConfig` file from the TASKING project directory into the configuration directory of the TESSY project. It is also recommended to copy the `OConfig` file found in TESSY's installation directory, i.e.

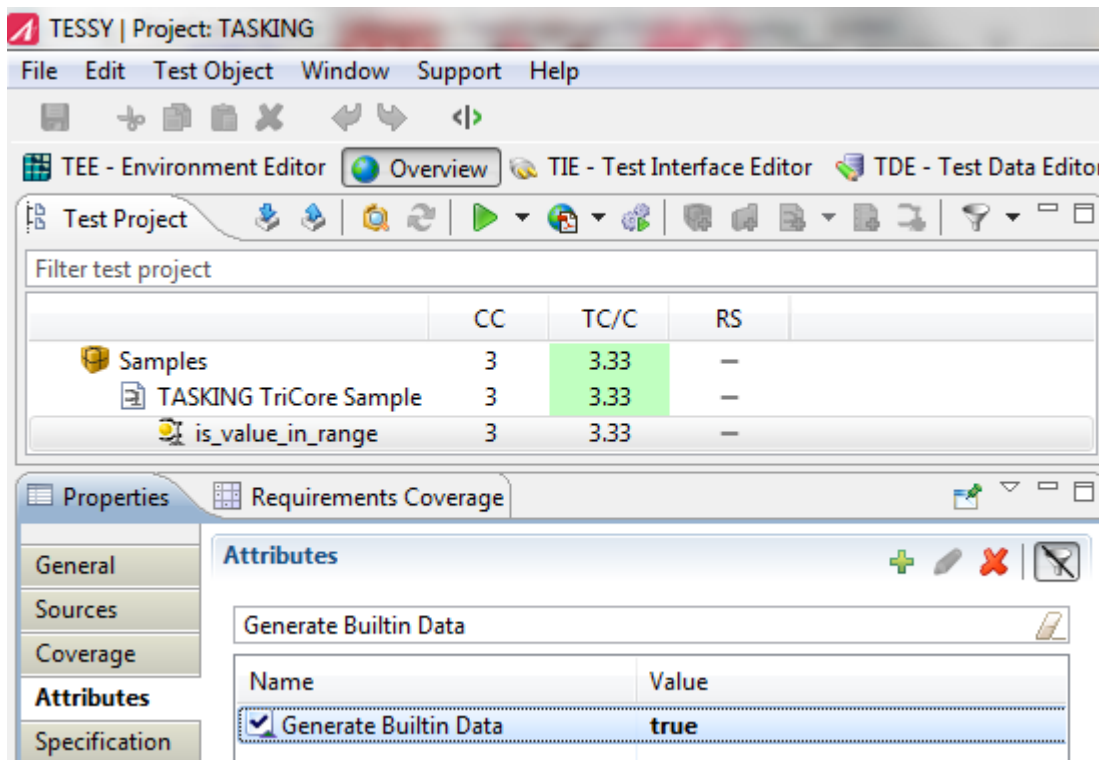
sys\targets\tasking\OConfig, into the same directory. The file contains a command for the TASKING debugger to disable the watchdog.


5 Interactive Debugging

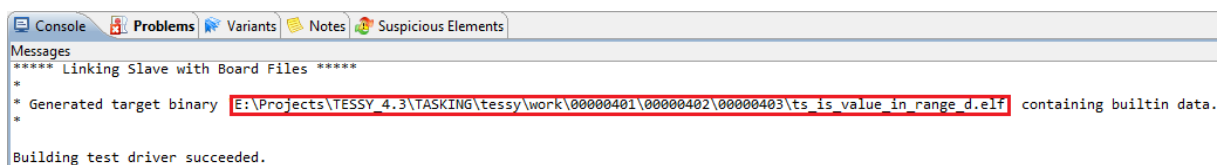
The TASKING VX debugger, on the other hand, is an Eclipse application of which the integrated debugger cannot be controlled from outside. Therefore, TESSY does not support the interactive debugging feature when executing tests. If you want to debug the test application with your test data, you need to create a test application with built-in test data.

5.1 Creating a test application to debug

Select your test object and go to the **Attributes** tab of the **Properties**. Set the **Generate Builtin Data** attribute to `true`.



Press the **Execute Test** button  to start compilation and linking of the test application containing the test data. At the end of this process, you should see the following output within the **Console** view:



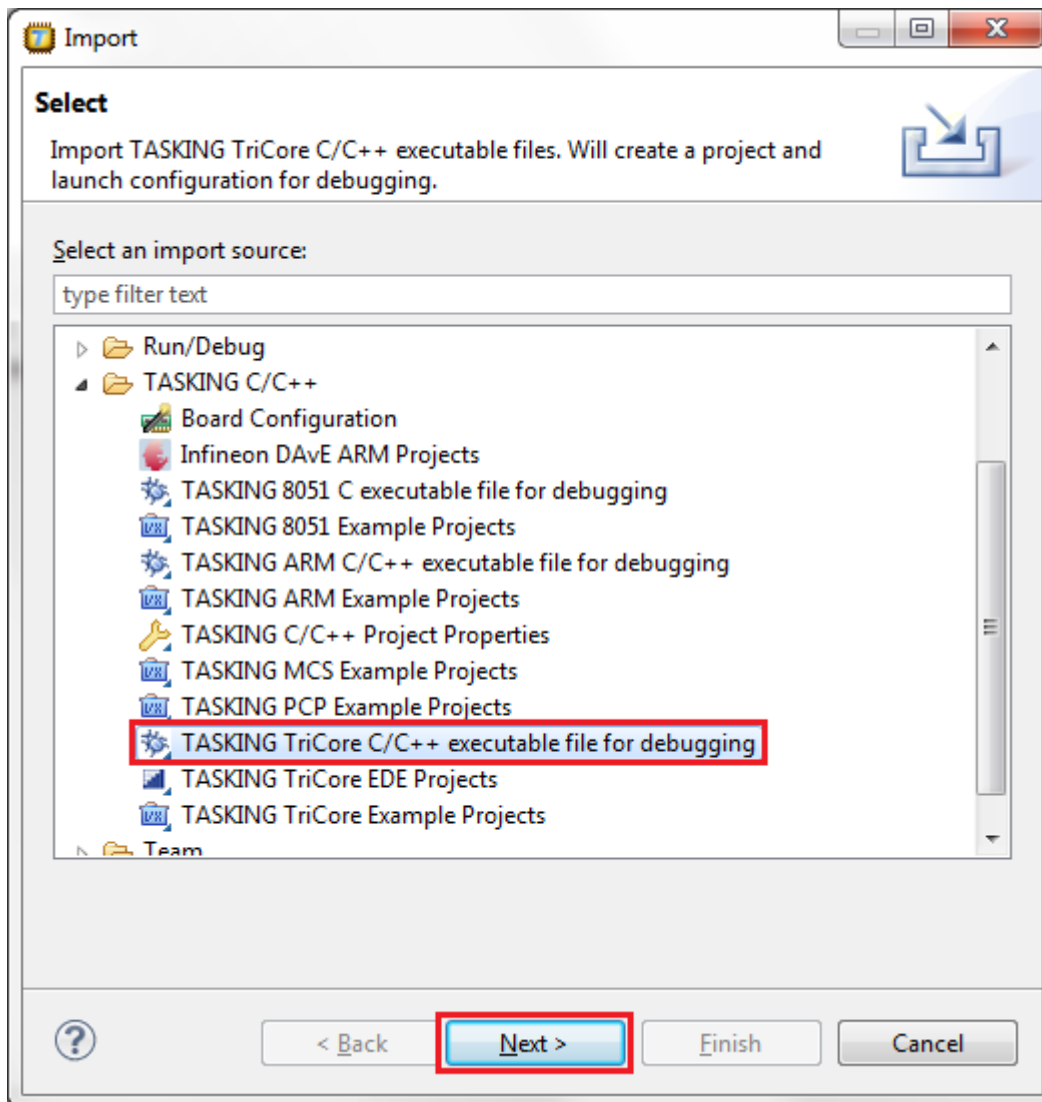
Copy the name of the generated executable ELF file into the Windows clipboard for the next step of debugging.

5.2 Debugging the test application

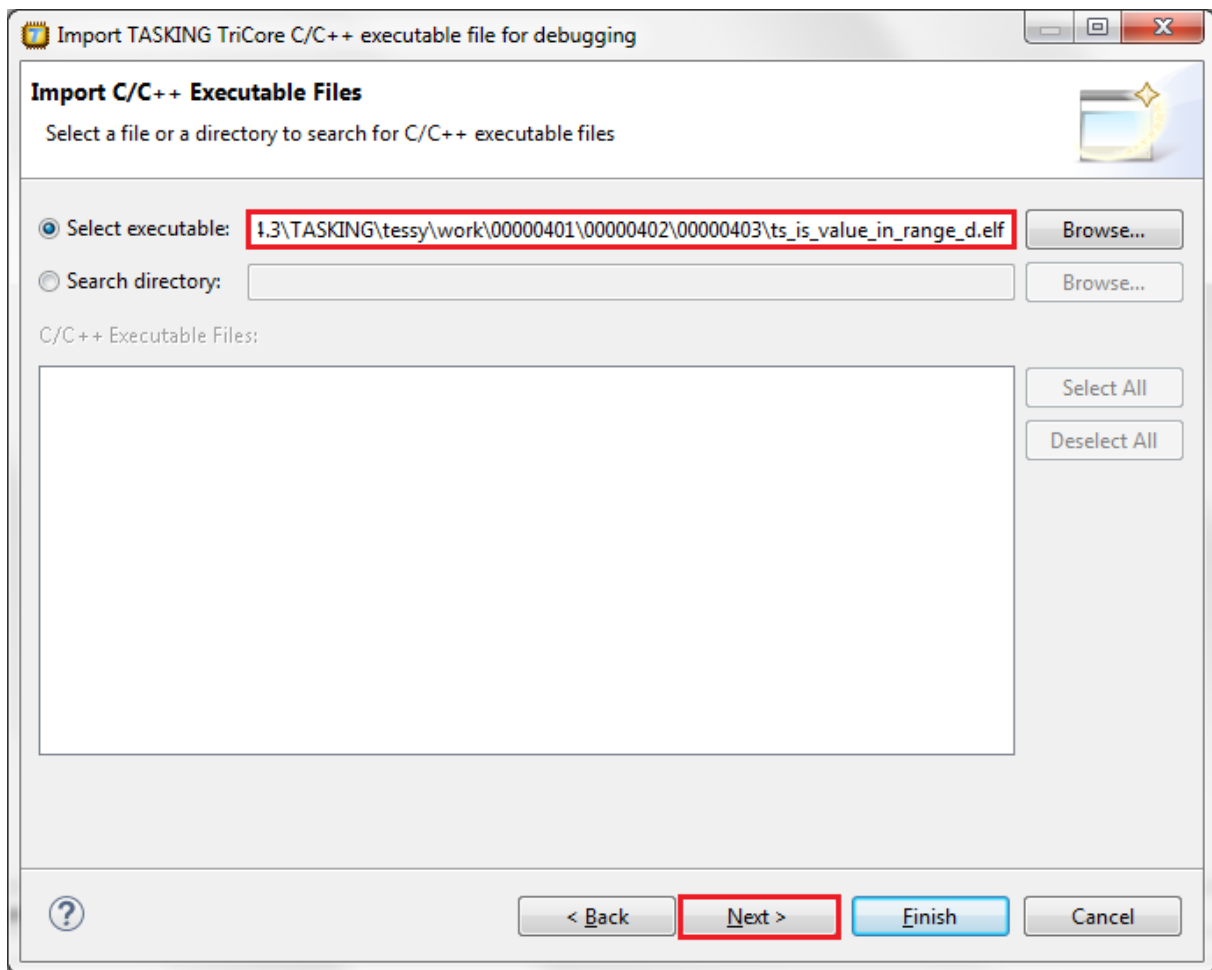
The debugging of the test application containing built-in data requires manual loading of this application into the TASKING EDE.

5.2.1 Importing the test binary

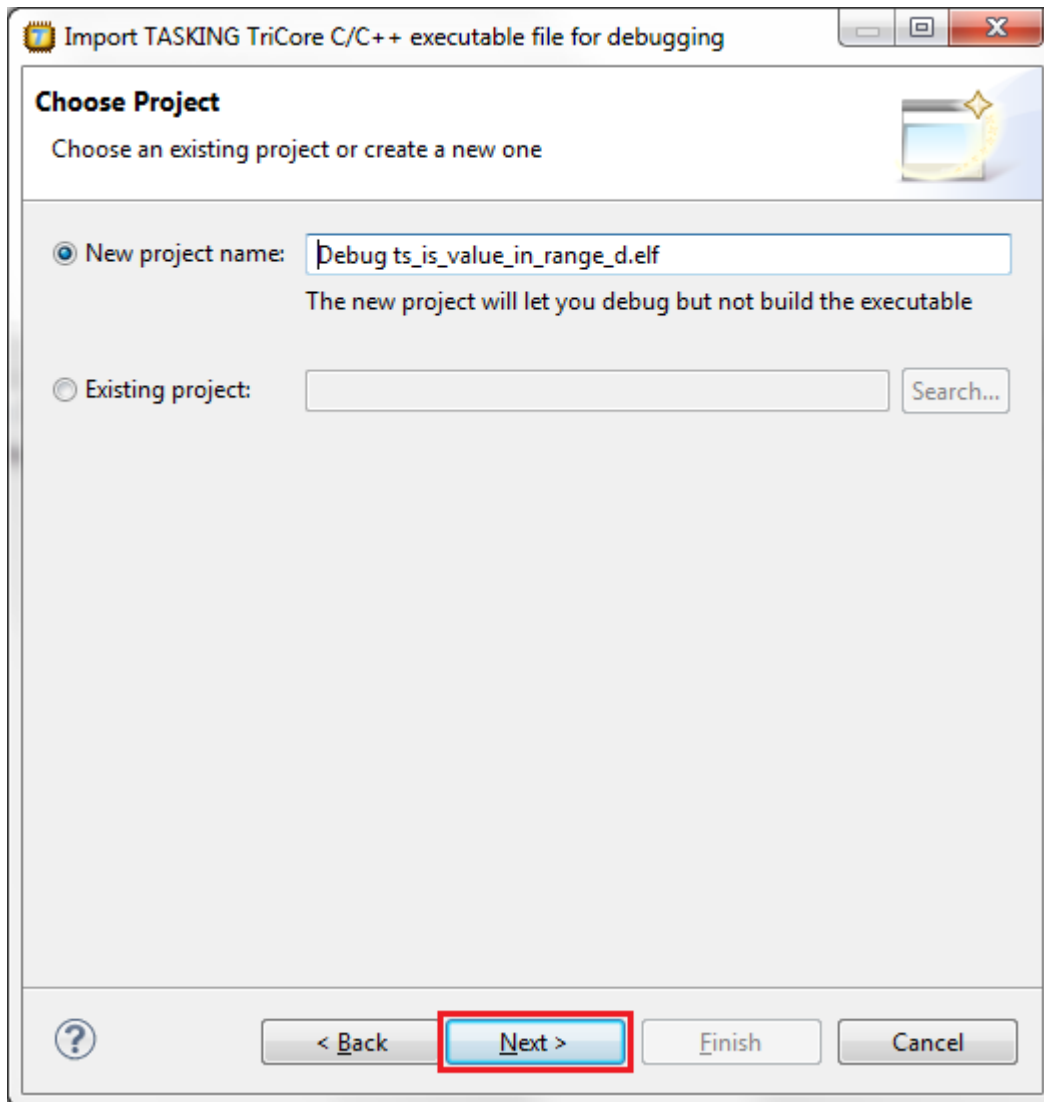
Select **File->Import...** to open the **Import** dialog:



Select **TASKING TriCore C/C++ executable file for debugging** and click **Next**.

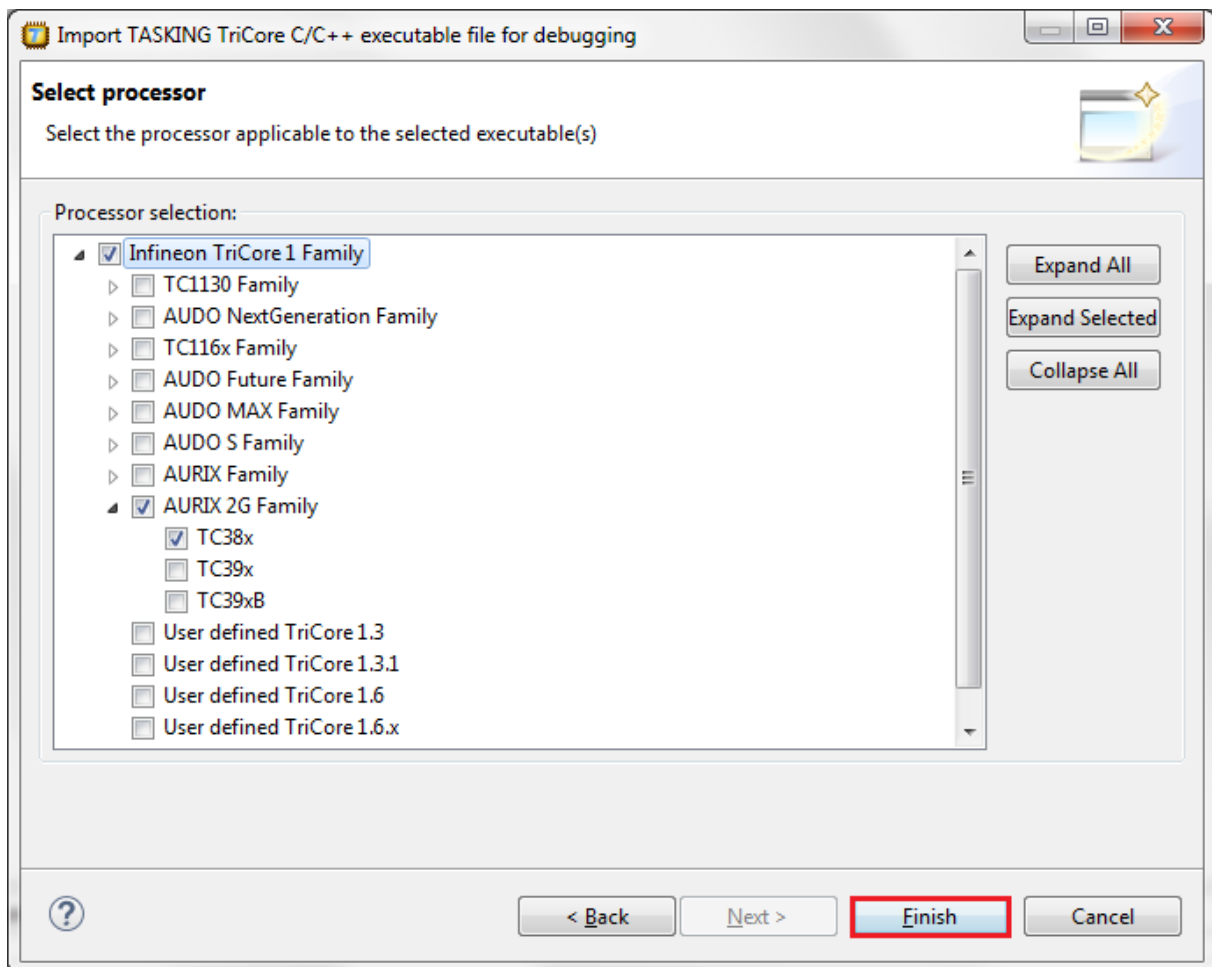


Enter the path to your test binary you copied from TESSY's **Console** view and click **Next**.



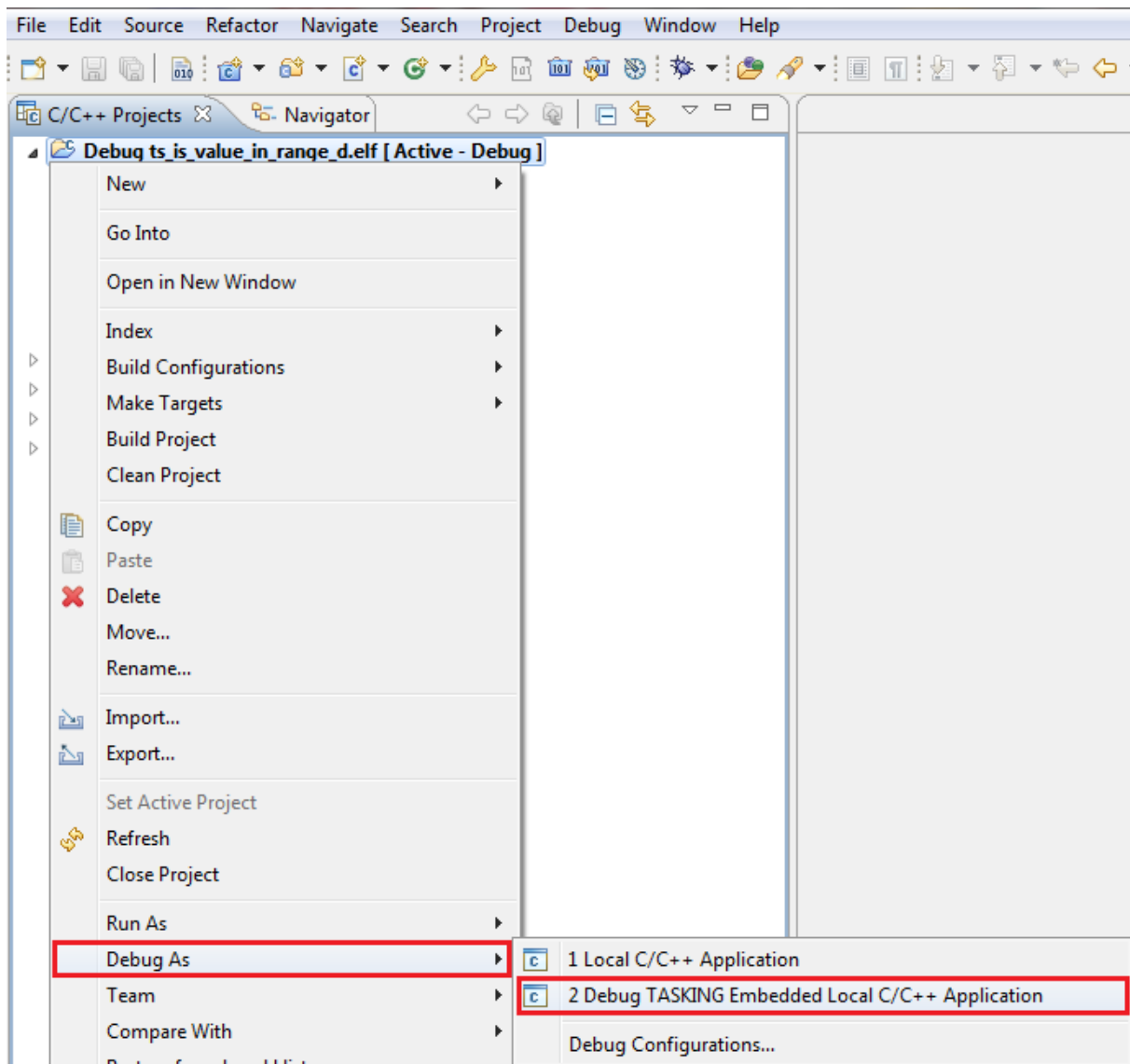
Click **Next**.

Select your target processor and click **Finish**.



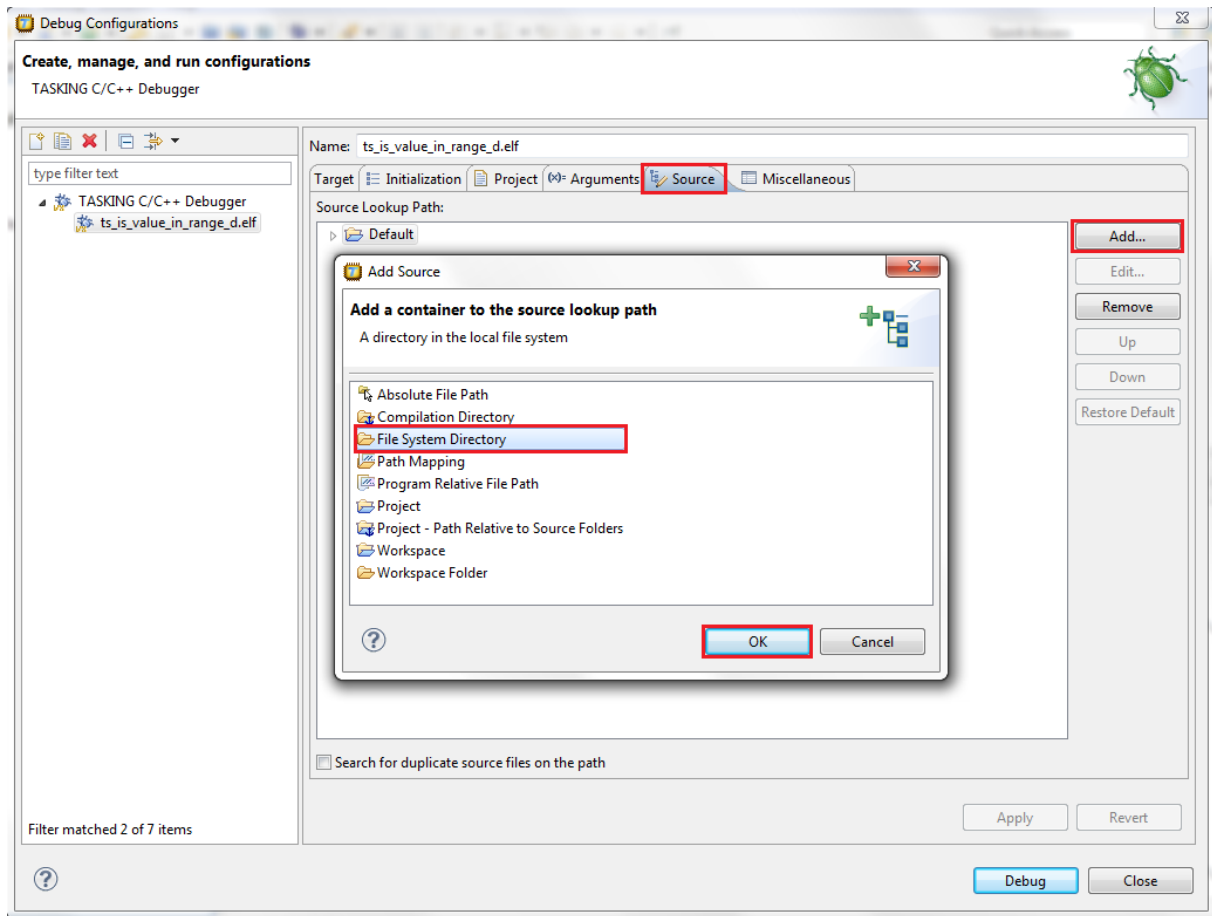
5.2.2 Debugging the test binary

Select your imported test binary project and open the context menu.

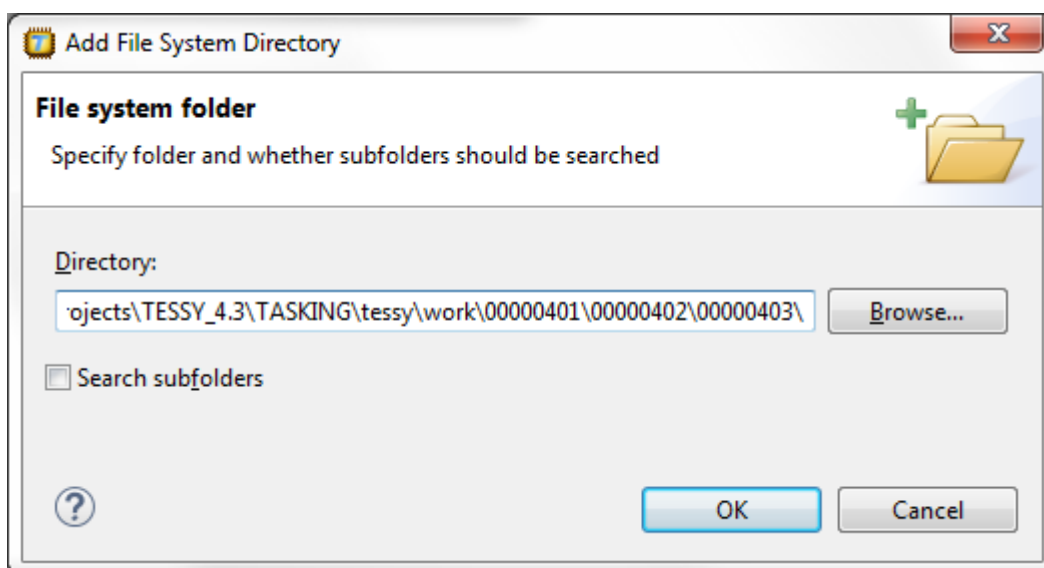


Select **Debug As->Debug TASKING Embedded Local C/C++ Application** to open the **Debug Configurations** dialog.

Switch to the **Source** tab and click **Add....** Select **File System Directory**.

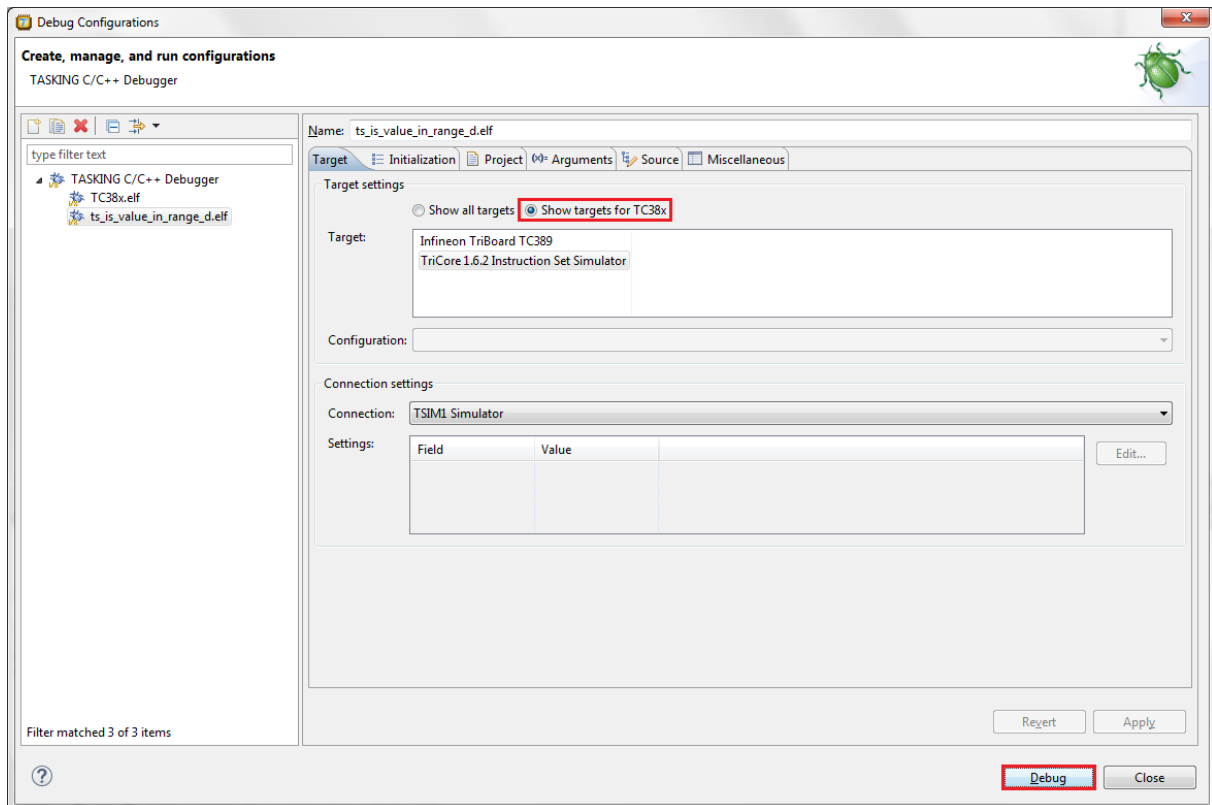


Add the directory of the test application into the input field and click **Ok**.

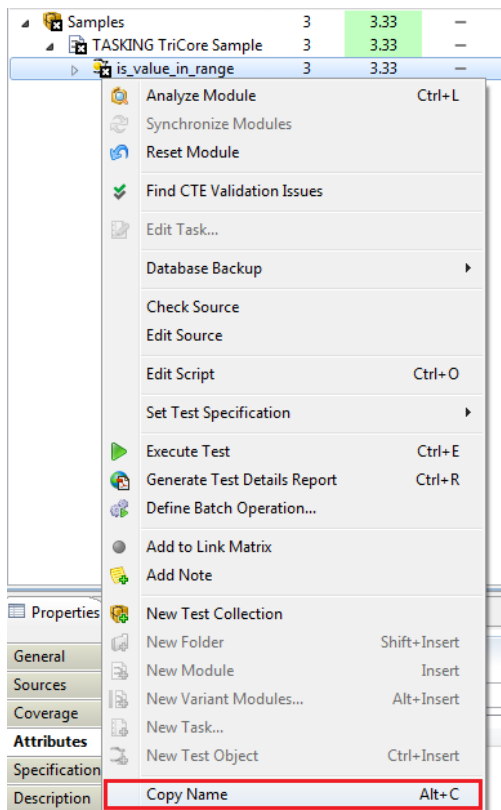


TESSY Application Notes

Switch to the **Target** tab, select the proper target from the **Target** pane, click **Apply**, and click **Debug**.

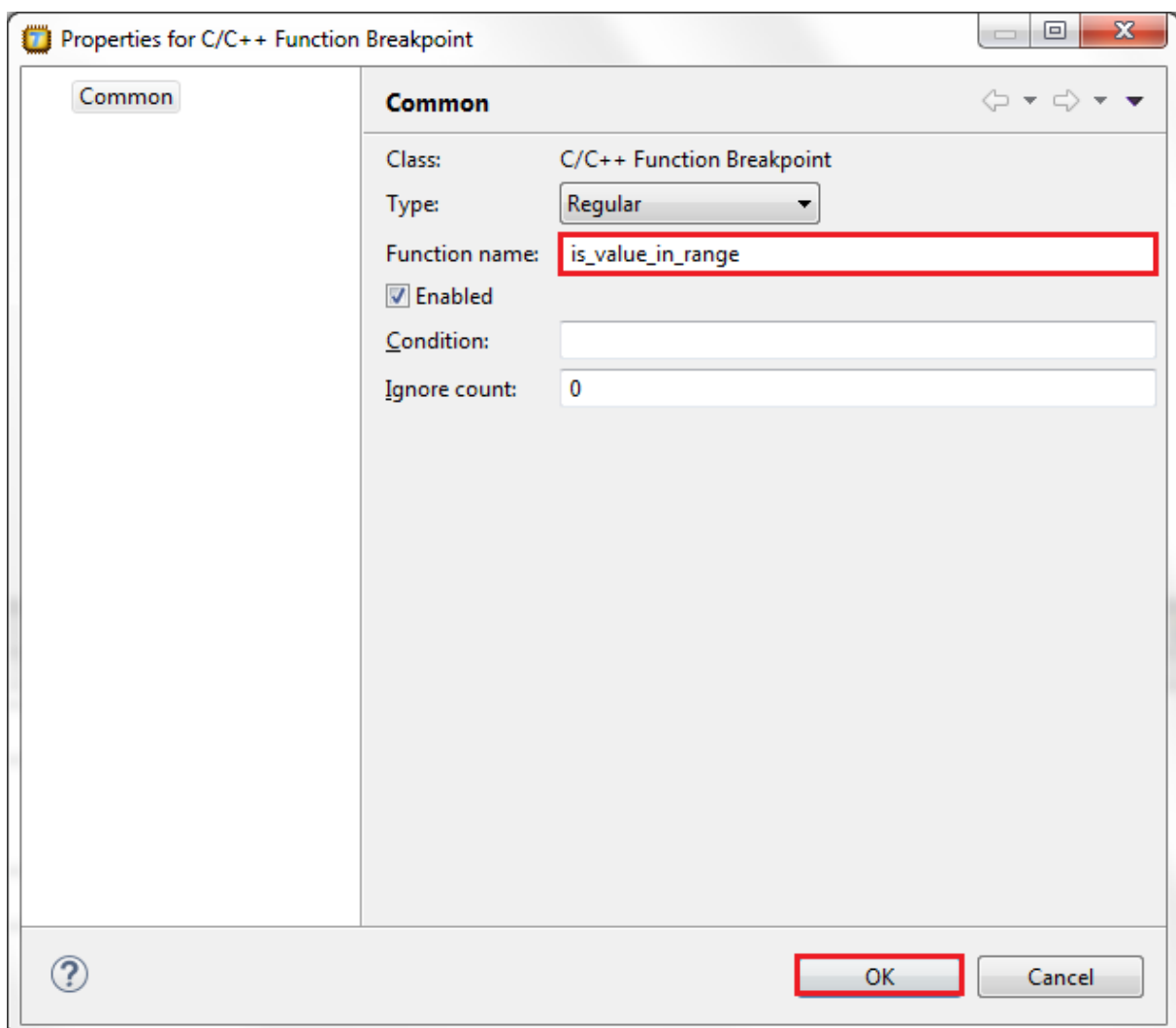
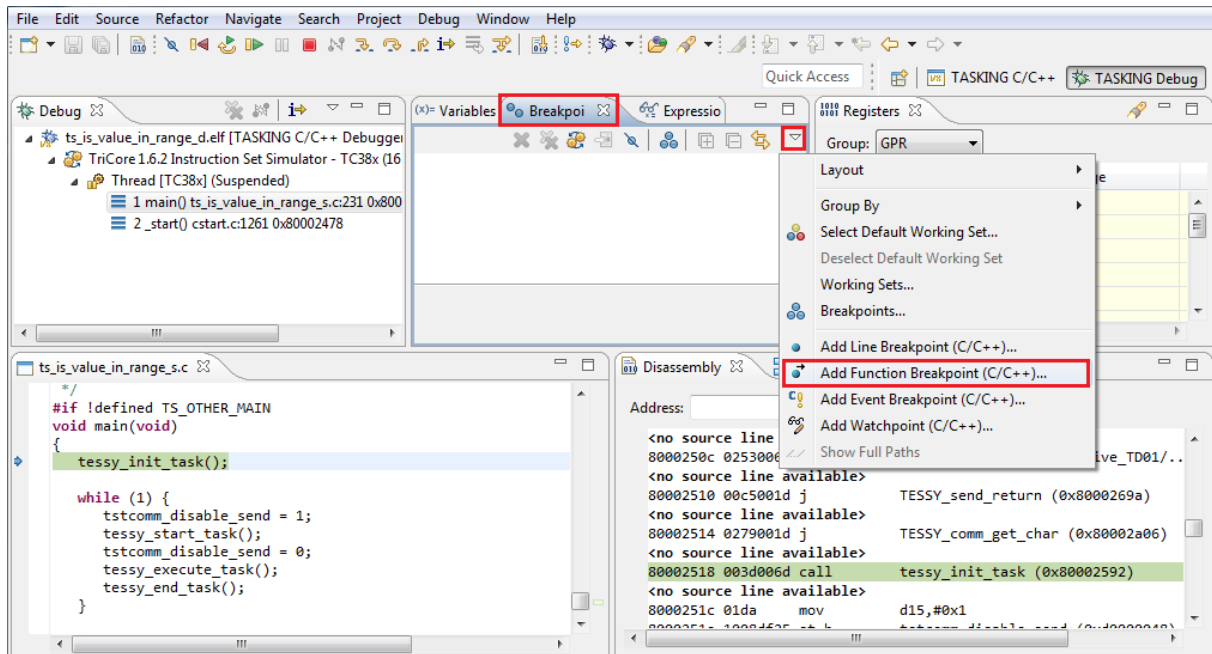


Copy the test object name from TESSY's **Test Project** view.



TESSY Application Notes

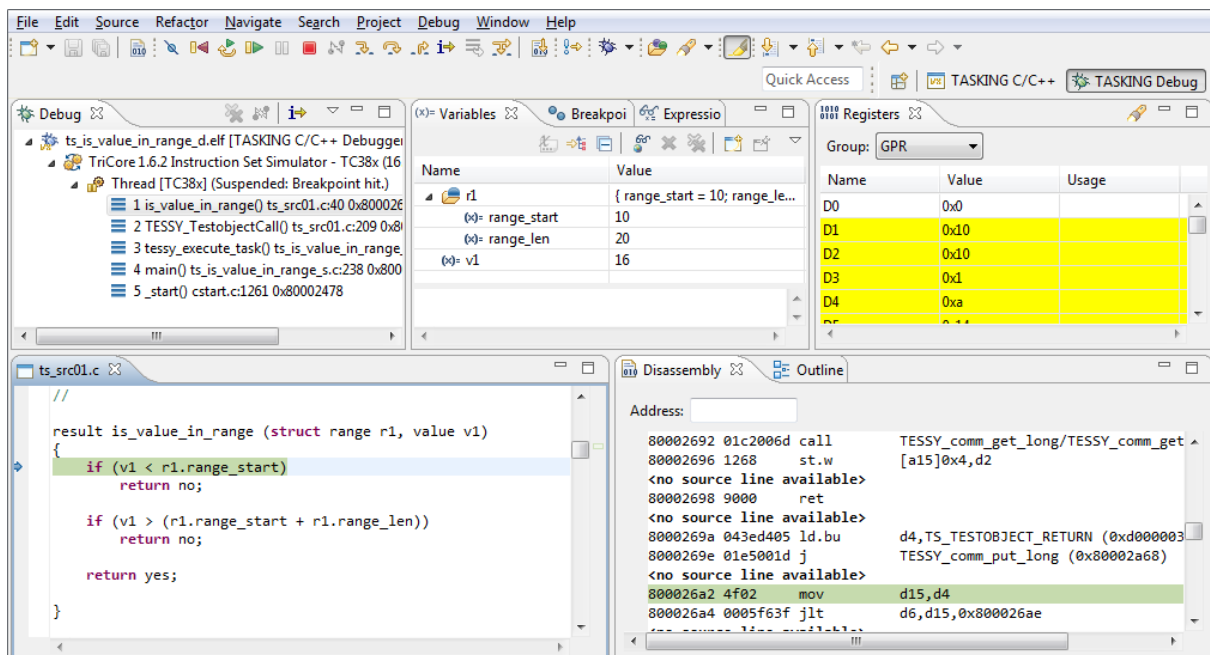
and add the name as a function breakpoint into the TASKING debugger.



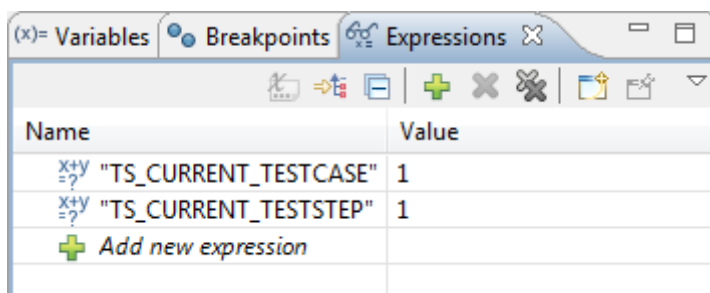
Resume the test execution.



You can now step through the function and continue test execution to run until the next test step. When you are finished, just terminate the debug session.



The test object contains two global variables which hold the current test case and the current test step which you may find helpful to be added to the **Expressions** view.



Please note: At the end of debugging do not forget to reset the **Generate Builtin Data** attribute of the current test object to `false` to enable normal test execution without debugging again.

6 Troubleshooting

6.1.1 The script debugger runs into a watchdog/trap

Please, copy the `OConfig` from the TESSY installation directory into your **Debugger Launch Path**. The file is found in `sys\targets\tasking\OConfig`.