

Texas Instruments Code Composer Studio (CCSv2/3 and CCSv4 or higher)

Abstract

This document describes the integration with the CCS debugger. Both compiler and debugger issues are described. The different versions of CCS require different setups as described below. TESSY's configuration setup for CCSv4 or higher is identical except for the TEE attribute **TessyCCSCmd**. For versions up to 5.2 of Code Composer Studio copy the contents of attribute **TessyCCSCmd v5.2** and paste it into attribute **TessyCCSCmd**. The default value of attribute **TessyCCSCmd** should be compatible with the latest version of Code Composer Studio.

Please note: Since version 4 of the CCS debugger the GUI is not available to debug tests interactively with TESSY. There is an alternative solution provided to create a test application having the test data built into the binary program. See 2.3.

Table of Contents

Abstract	1
1 Introduction.....	3
2 CCSv4 or Higher	3
2.1 CCS Target Configuration	3
2.1.1 Step 1: Create a new workspace	3
2.1.2 Step 2: Create a new project.....	3
2.1.3 Step 3: Create a target configuration	5
2.1.4 Step 4: Enter the target configuration within TEE.....	7
2.1.5 Multi-core initialization.....	8
2.2 Executing a Test Run	9
2.3 Debugging Using the CCS GUI (with built-in Test Data)	10
2.3.1 Step 1: Change the module properties.....	10
2.3.2 Step 2: Run the test	11
2.3.3 Step 3: Start CCS debugger.....	12
2.3.4 Step 4: Load the test application with built-in test data	12
2.3.5 Step 5: Set a breakpoint.....	14
2.4 Troubleshooting When Debugging	16
2.4.1 Reloading the test application	16
3 Using TI ARM CLANG Compiler	16
4 CCSv2/CCSv3.....	18
4.1 Files and Settings.....	18
4.1.1 Module attribute settings	19
4.1.2 Project file	20
4.1.3 Adapting the GEL startup file	20
4.2 Running the Test.....	21
4.3 Compiler Specific Settings	22

1 Introduction

This document describes the steps necessary to use the CCS debugger in conjunction with TESSY. There are completely different approaches for CCSv2/CCSv3 and the latest versions CCSv4 or higher. Please make sure to peruse the corresponding chapter for the CCS version you are using.

2 CCSv4 or Higher

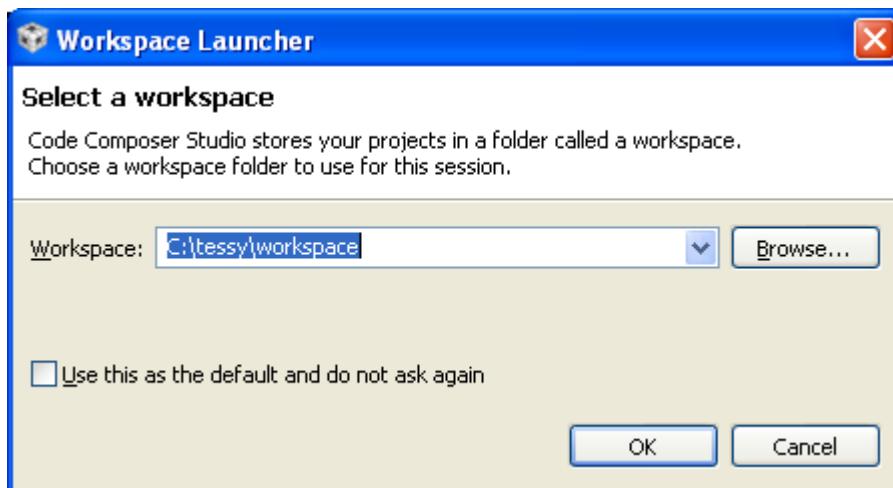
This chapter describes the steps needed to use TESSY in conjunction with Code Composer Studio 4 or higher (CCS) as target debugger. Since version 4 Texas Instruments introduced a new IDE including also a new debugger API. The setup dialogs may differ between the different versions of Code Composer Studio from version 4 up to version 11, which is the latest one we have tested by now.

2.1 CCS Target Configuration

In order to use the CCS you need to create a target configuration file within CCS that will be used by TESSY.

2.1.1 Step 1: Create a new workspace

If you do not already have a workspace for your development project, you need to create a new one to work with CCS. You will be asked for a workspace when starting CCS.

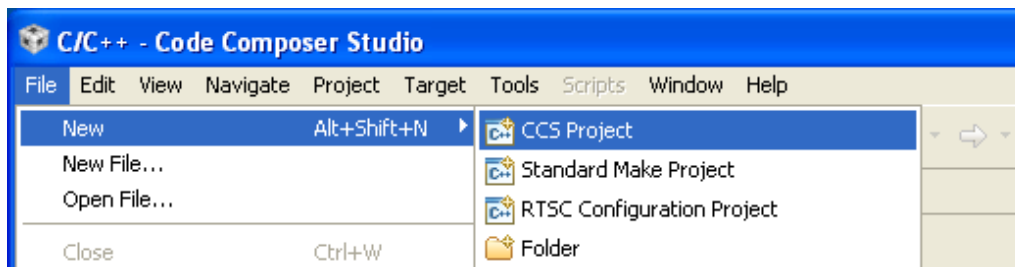


Select an appropriate location for your workspace.

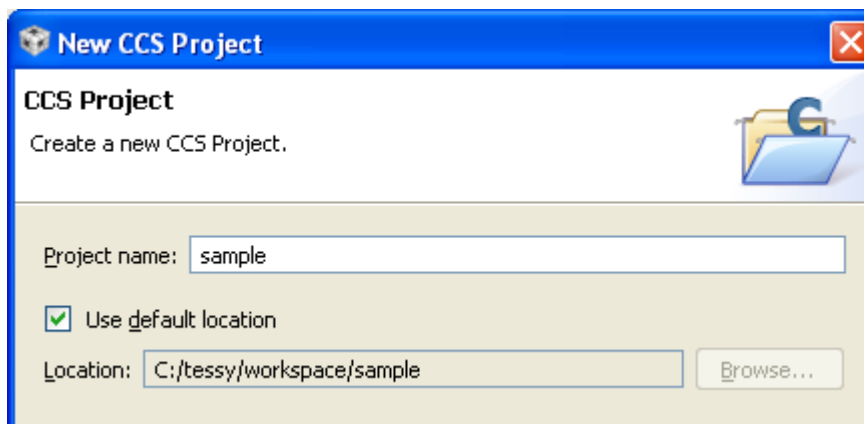
2.1.2 Step 2: Create a new project

Create a new project within CCS to be used for testing the target configuration we will prepare for TESSY. You may also use your development project (if available).

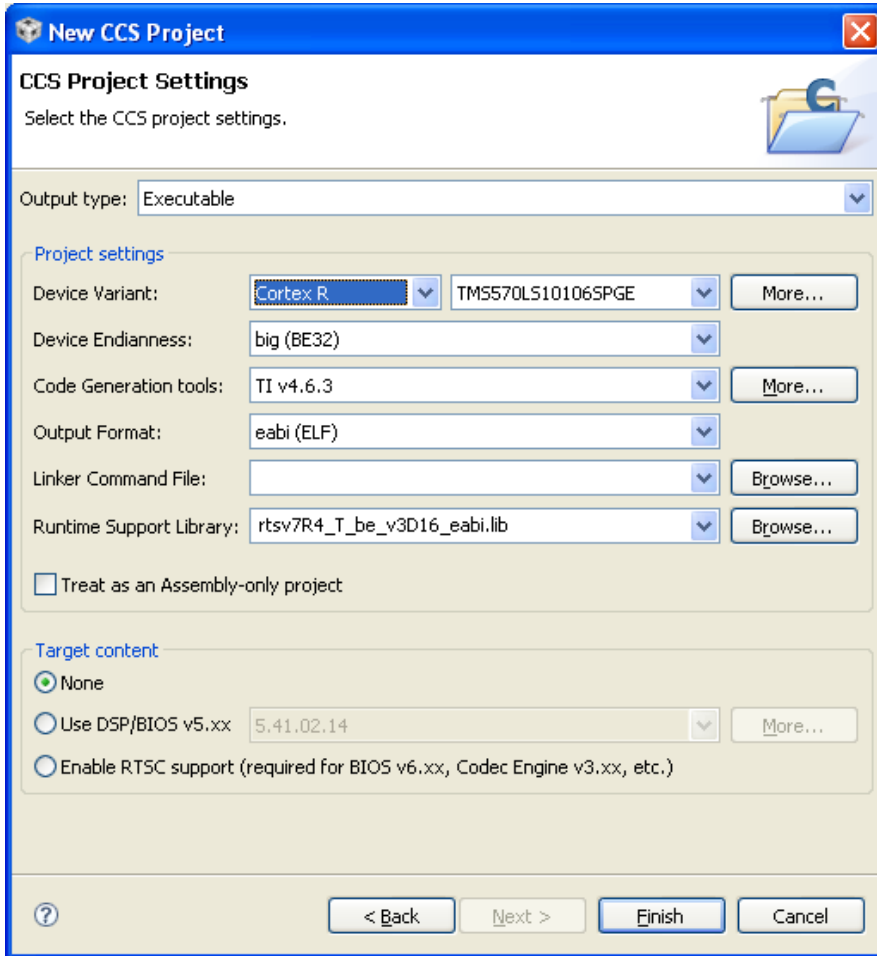
In order to create a new project, select **New->CCS Project** from the **File** menu.



This will bring up the **New CCS Project** dialog.



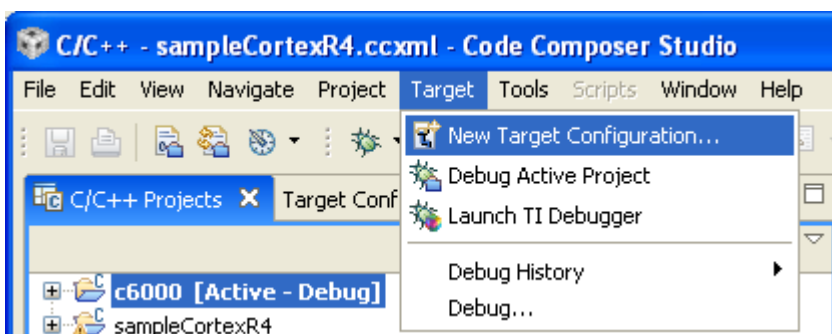
Enter “sample” as project name and work through the rest of the CCS project creation wizard. Refer to the CCS help menu for help on any CCS settings. The following shows for example the settings for the TMS 570 Cortex R controller:



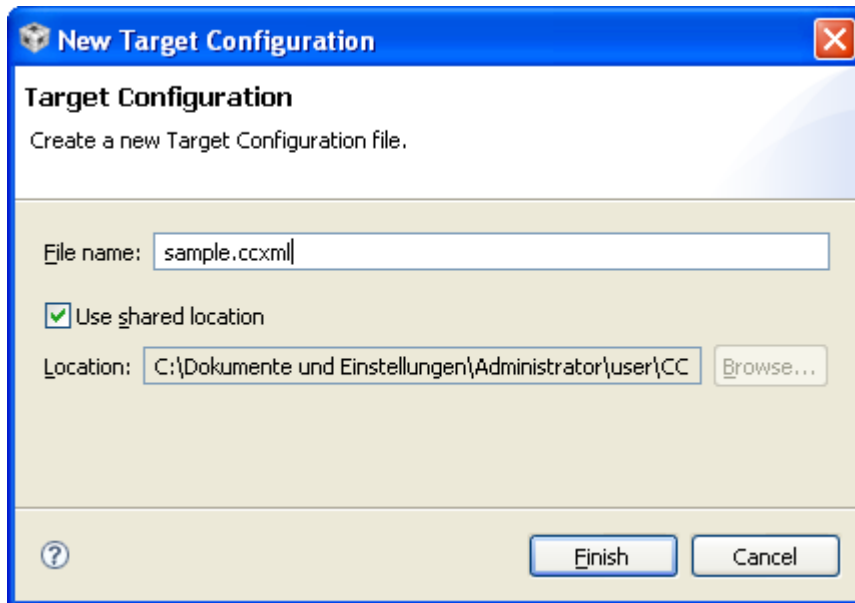
When clicking the **Finish** button, the new project will be created. Now add at least one source file and compile the project within CCS.

2.1.3 Step 3: Create a target configuration

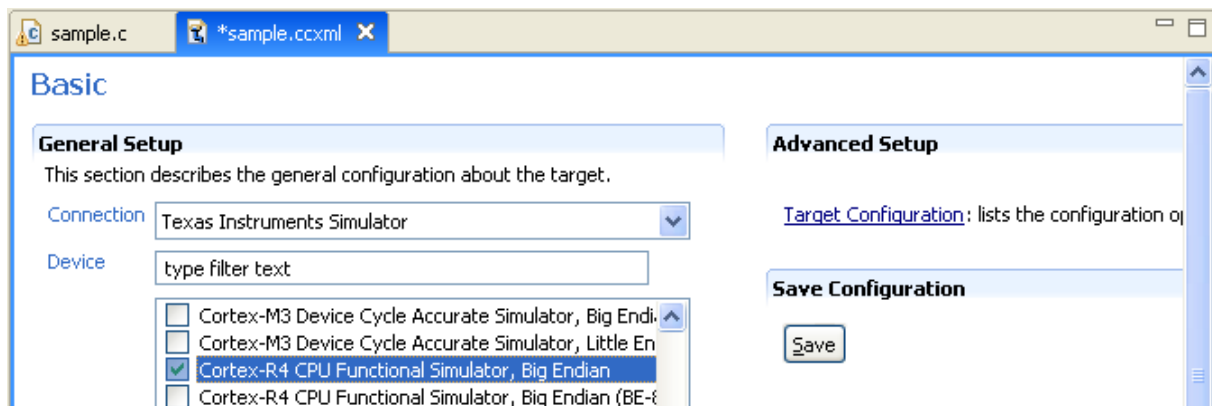
The execution of the project binary requires a target configuration within CCS. Either use any existing CCS configuration (e.g. from your development project) or create a new one as described below. Select **New Target Configuration...** from the **Target** menu of CCS.



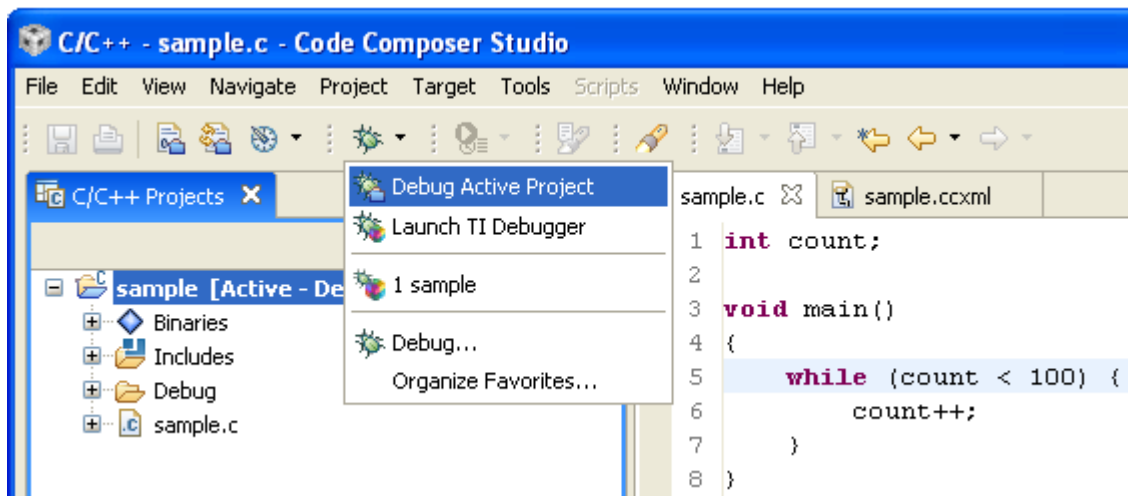
This will open the **New Target Configuration** dialog. Please select a name for your configuration file (e.g. "sample.ccxml").



Apply all required settings within the target configuration (e.g. select the **Cortex-R4** simulator) and press the **Save** button.



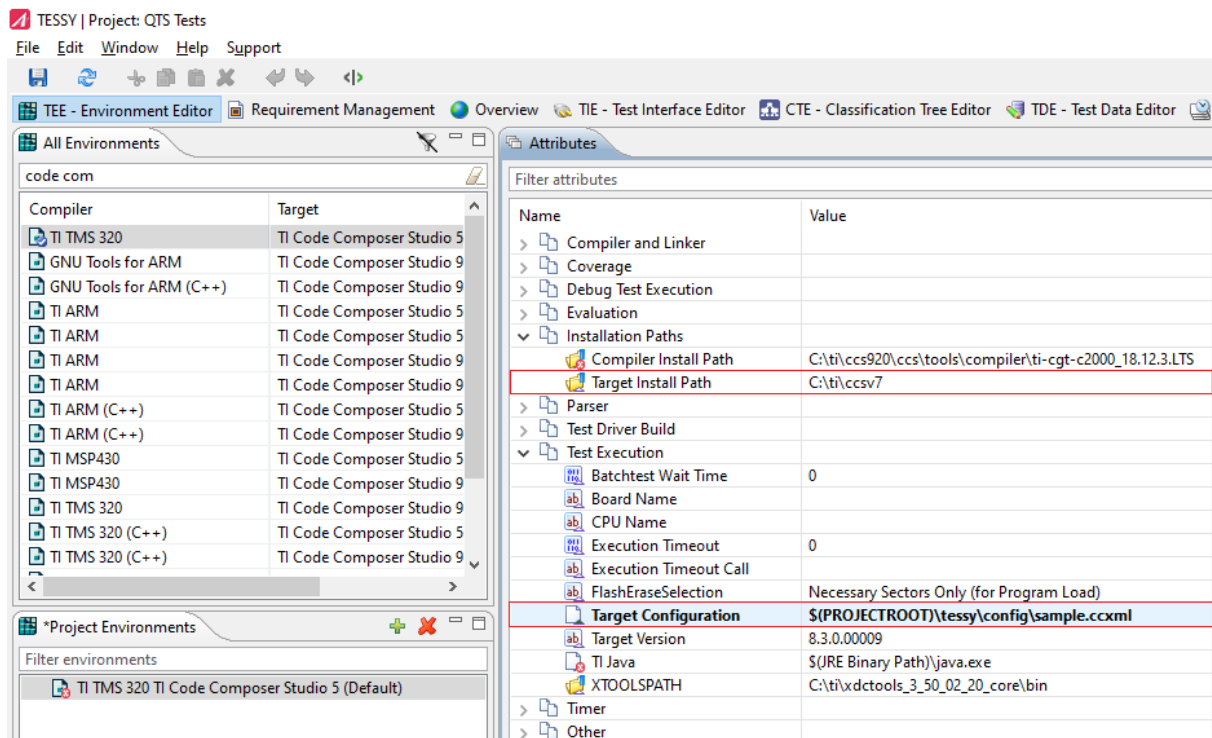
The newly created target configuration will be the default target configuration within CCS. Now run your sample project with the new target configuration.



If everything works fine, you may use the CCS target configuration file with TESSY.

2.1.4 Step 4: Enter the target configuration within TEE

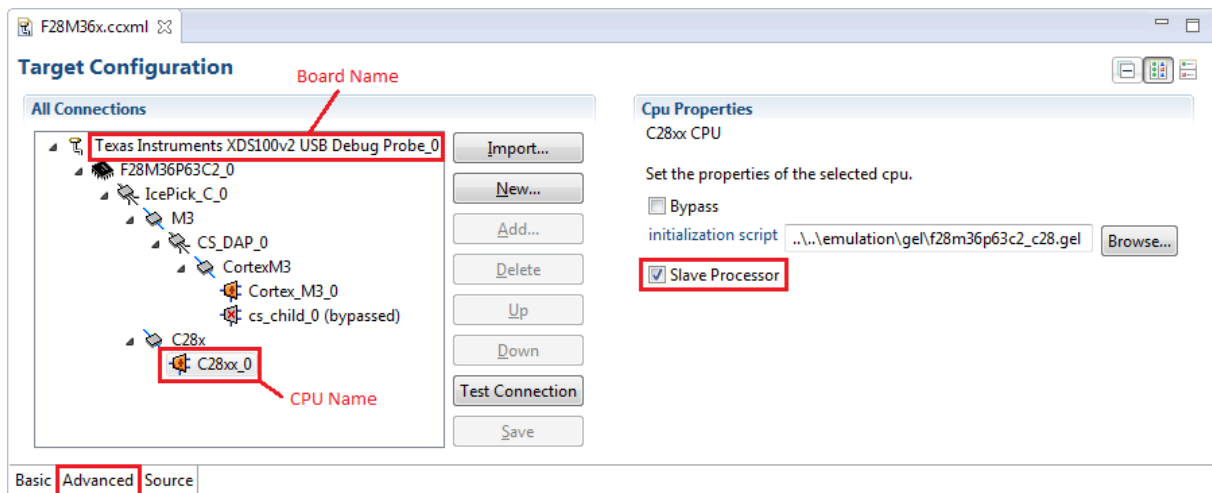
Start the TESSY Environment Editor (TEE) and change the **Target Configuration** attribute to the target configuration file you just created/edited.



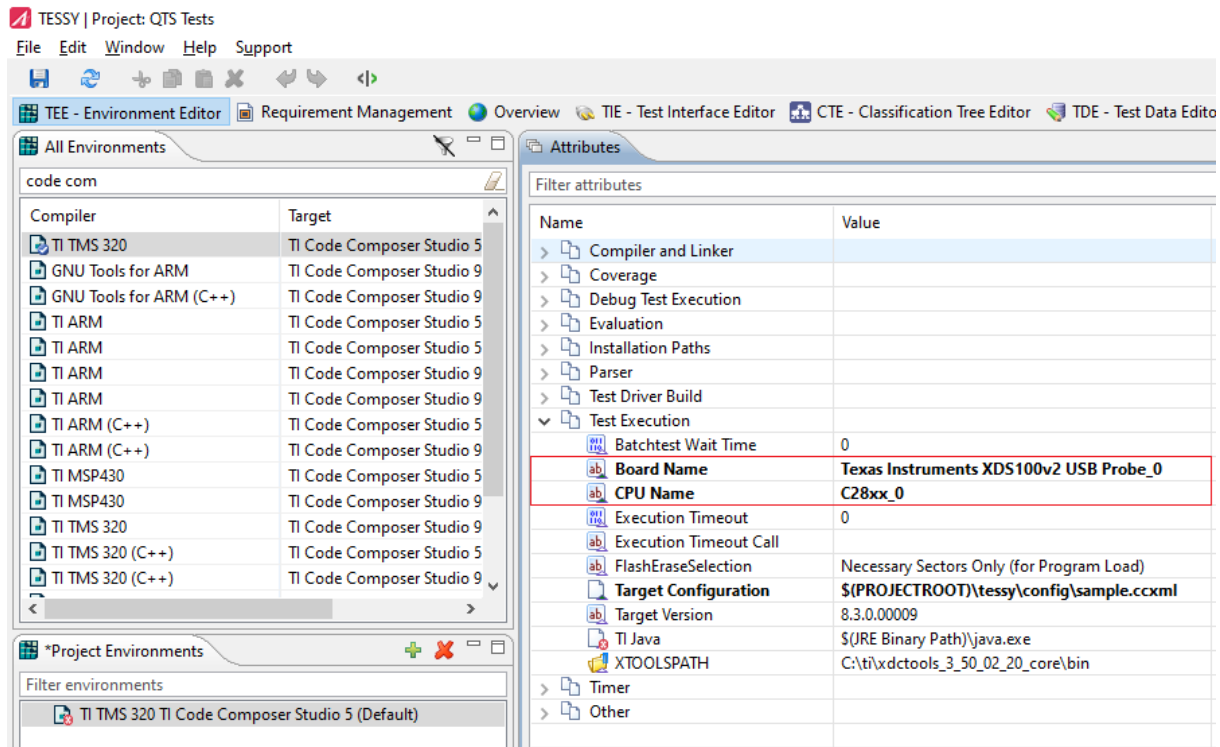
Also make sure that the **Target Install Path** points to a valid path where your CCS installation is located. Save and exit the TEE.

2.1.5 Multi-core initialization

For some environments it might be necessary to initialize one CPU and let the test binary run on a second one thereafter. In order to specify which board and CPU has to be used TESSY provides the attributes **Board Name** and **CPU Name**. You may obtain these values from Code Composer Studio's target configuration as demonstrated below.



The CPU name should point to the CPU which will be used for the test run, which for these cases is the slave processor. Set these values in the corresponding string type attributes of TESSY's environment editor as shown below.



2.2 Executing a Test Run

TESSY executes tests with CCS using the Java interface of CCS without running the GUI of CCS. Upon start of the test run, TESSY will start the TESSY DSS adapter (i.e. the connection to CCS) automatically. It will also stop the adapter appropriately, e.g. when TESSY is being closed.

Please note: When test runs are aborted from within TESSY, there may remain some Java processes in idle mode. These processes will be terminated when TESSY is being closed.

After the test execution you will see the test results within TESSY. If some test cases failed you may want to debug the test application. Please follow the steps described within the next chapter in order to build and debug the test application with the CCS debugger.

2.3 Debugging Using the CCS GUI (with built-in Test Data)

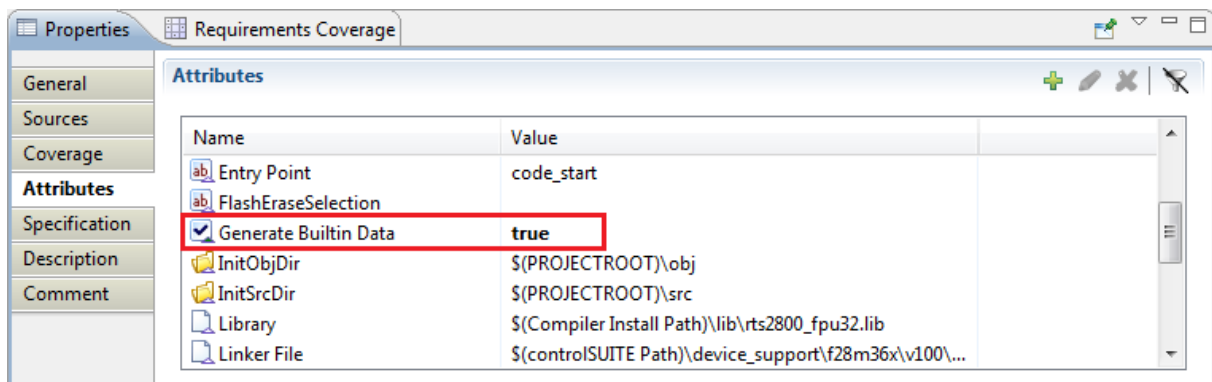
Since CCS does not support interactive debugging controlled from outside (i.e. from TESSY when executing tests), there is an option that allows creating a standalone test application which contains the test data already built-in. This scenario is available by switching a TESSY module attribute as described below. You may then load the generated test application into CCS and step through the code.

Please note: For testing in both execution scenarios, the TI debugger has to use exactly the same configuration file as you specified within the attribute **Target Configuration** within the TESSY Environment Editor (TEE).

2.3.1 Step 1: Change the module properties

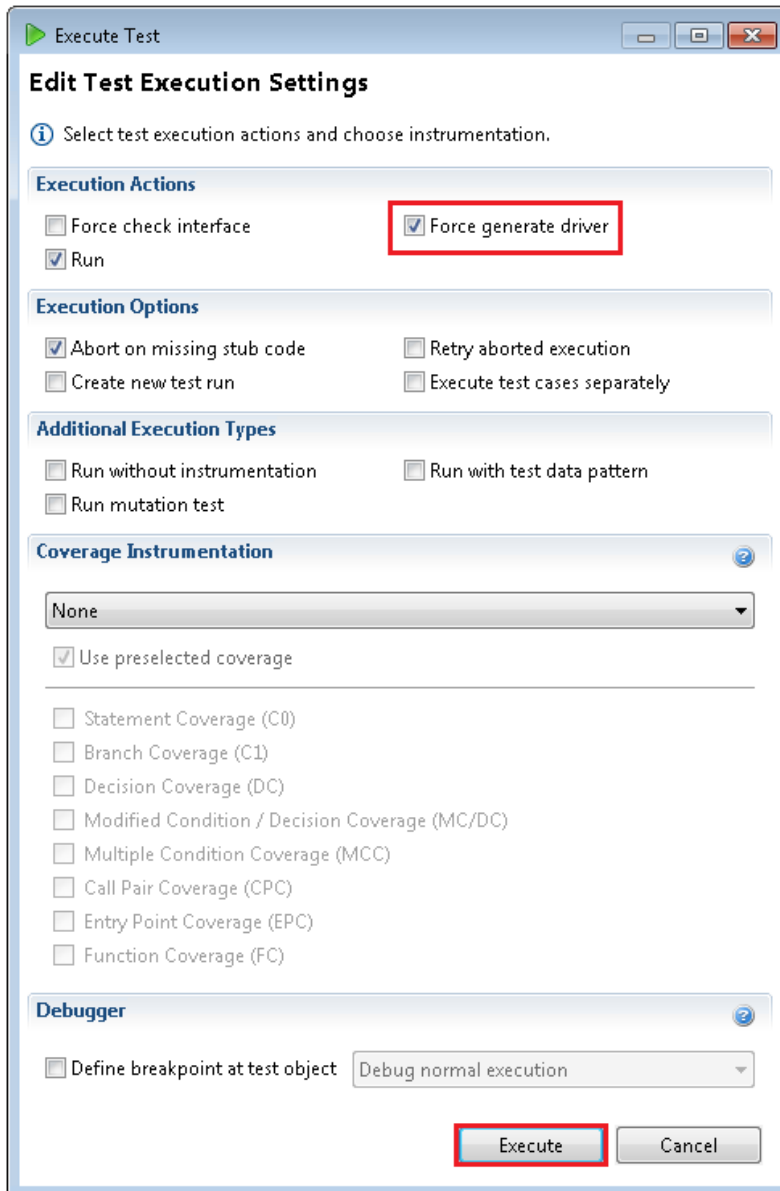
In order to generate a test application with built-in test data, select your test module and choose the **Properties** view. Select the **Attributes** tab and change the setting of the module attribute **Generate Builtin Data** to **true**.

Please note: Attribute **Comm Checksum** must not be set to **true** if **Generate Builtin Data** is set to **true**.

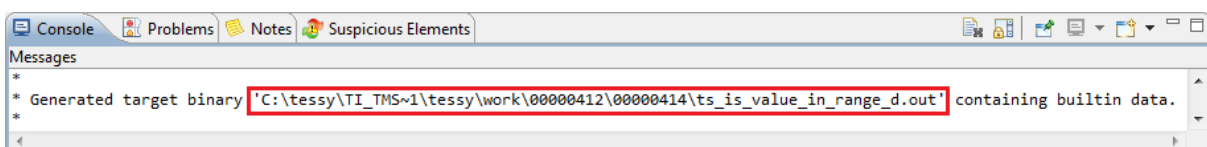


2.3.2 Step 2: Run the test

Start the test within TESSY with the **Force Generate Driver** option selected within the **Execute Test** dialog as shown below:



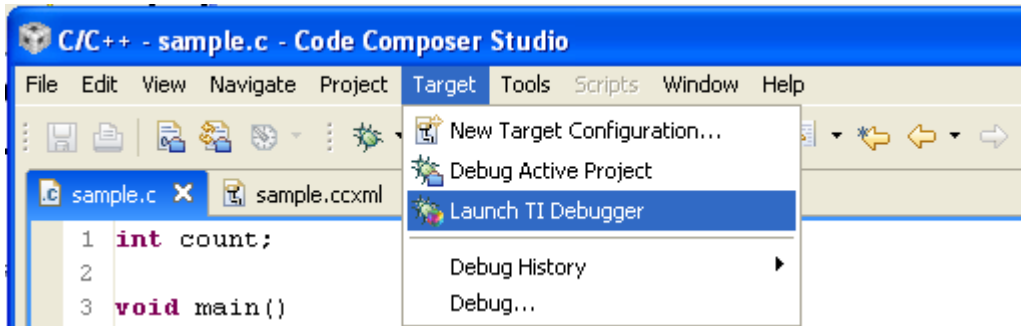
After pressing **Execute**, TESSY will build the test application with the test data built-in. The name and path of the generated test application will be shown within TESSY's **Console** view.



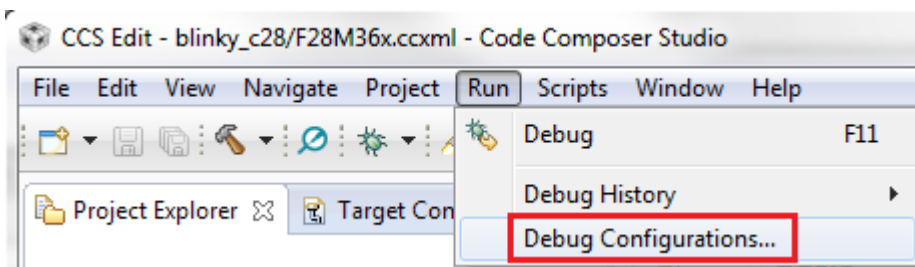
Now copy the file name of the test application.

2.3.3 Step 3: Start CCS debugger

Start the CCS debugger by selecting **Launch TI Debugger** from the **Target** menu

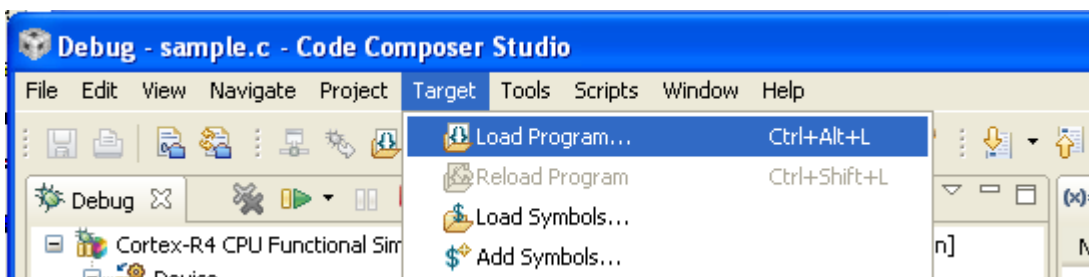


respectively select **Debug Configurations...** from the **Run** menu.

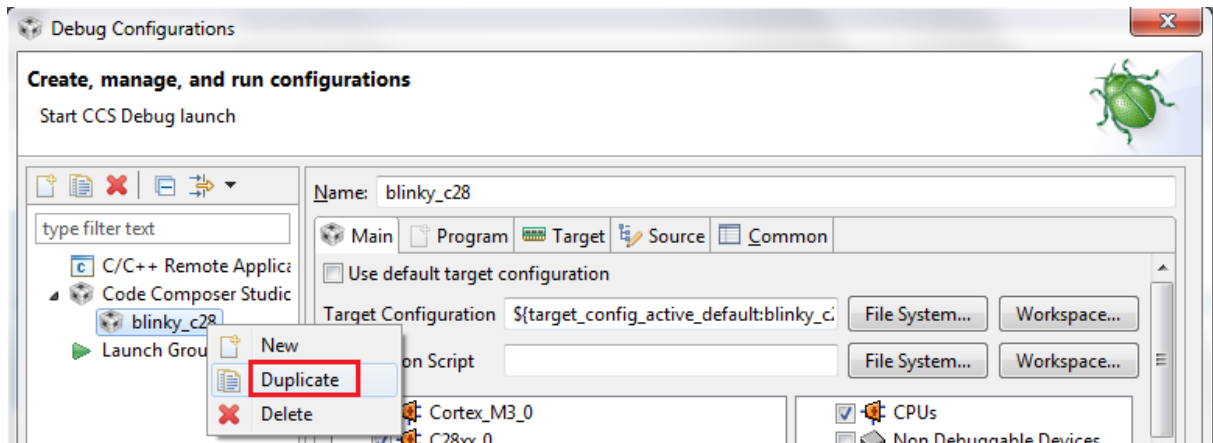


2.3.4 Step 4: Load the test application with built-in test data

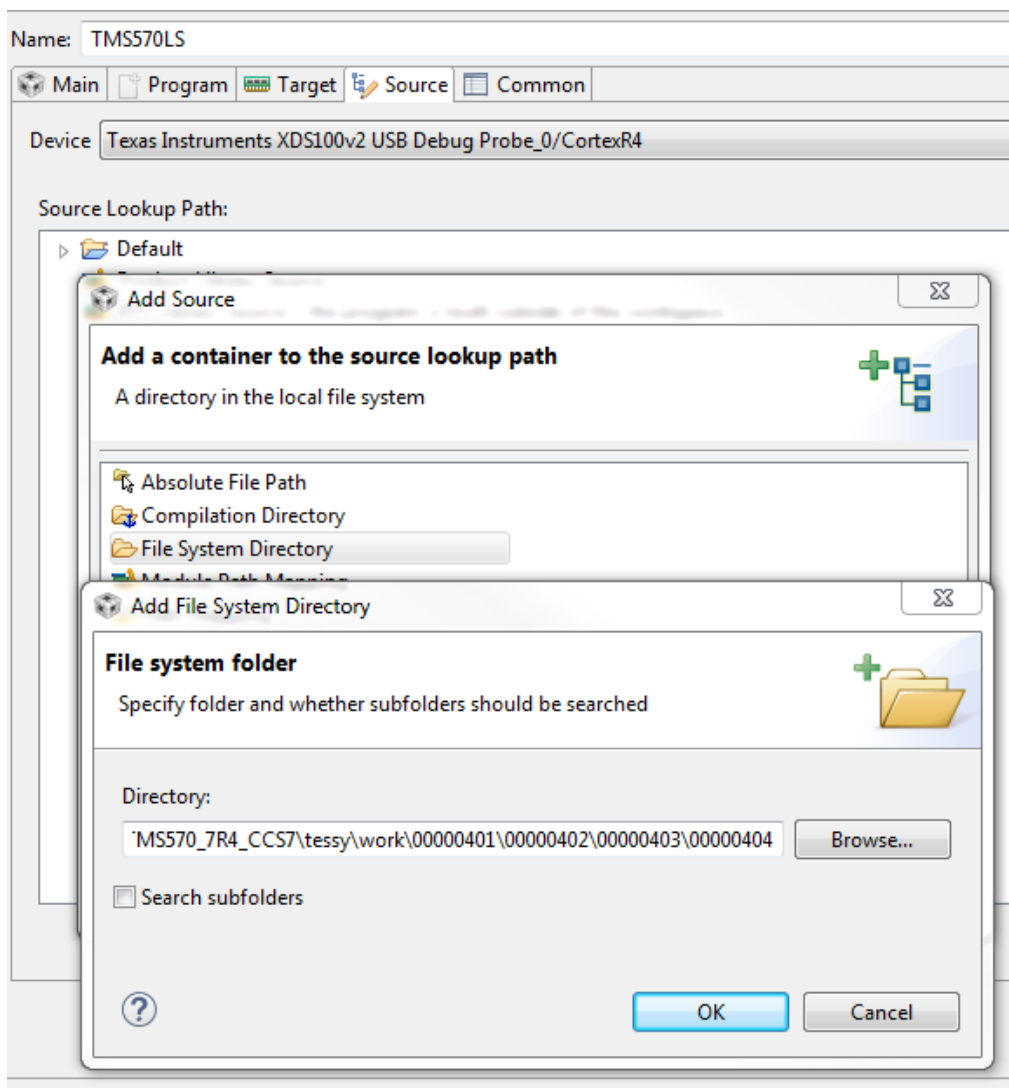
Load the generated binary into CCSv4 (or higher) using **Load Program...** from the **Target** menu

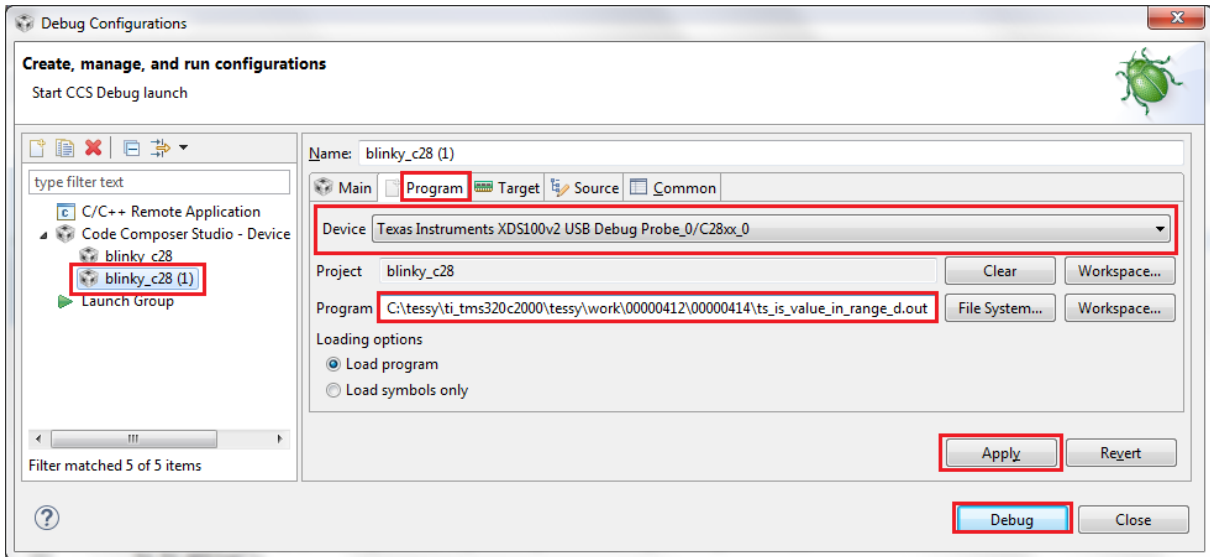


respectively duplicate your own debug configuration,



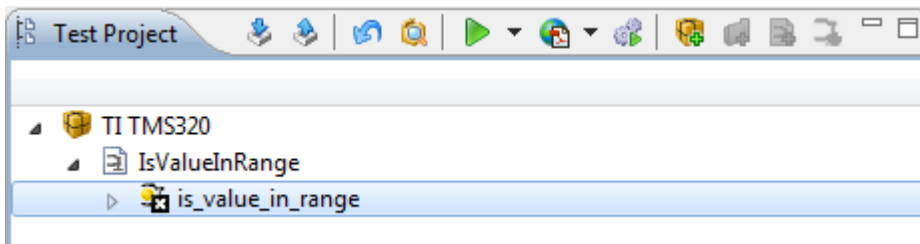
select the **Program** tab, choose the device, paste the copied path of your test application, click **Apply**, select the **Source** tab, click **Add...**, choose **File System Directory**, paste the directory of the test application, click **Apply** again, and click **Debug**.



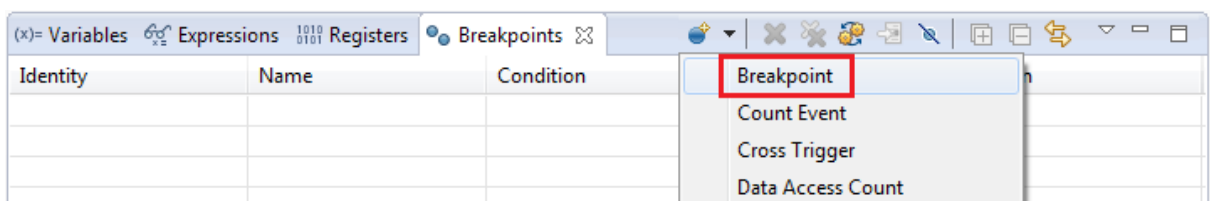


2.3.5 Step 5: Set a breakpoint

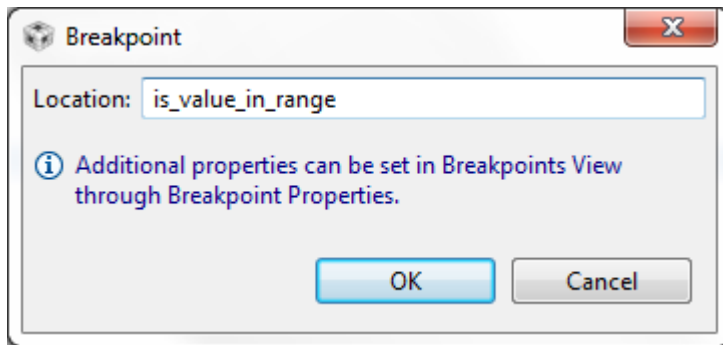
In order to stop at your test object, you need to set a breakpoint. The easiest way to do this is to switch back to TESSY and select the test object.



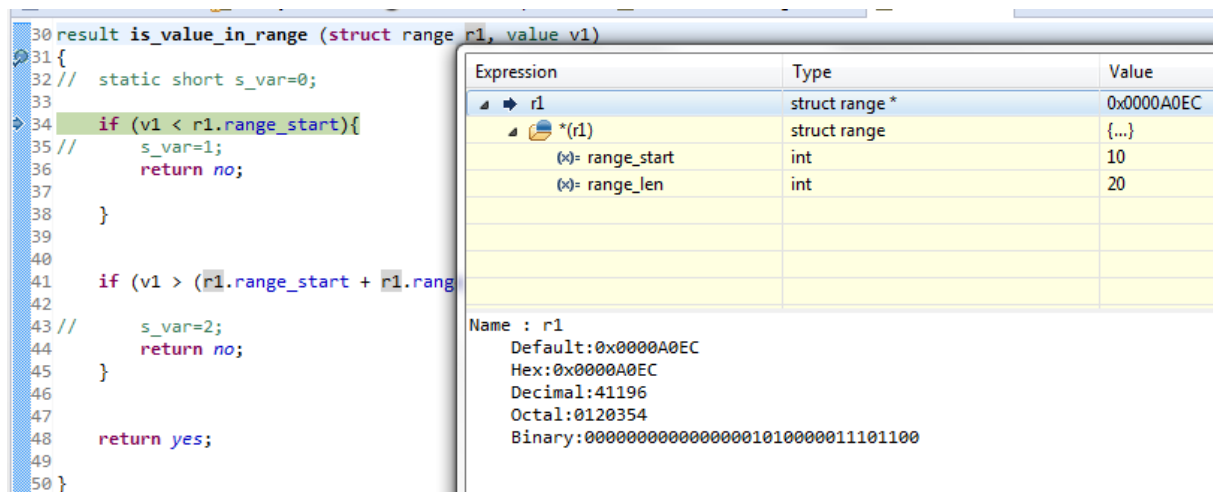
Now press **Alt-C** to copy the name of your test object and switch back to CCS's **Breakpoints** view (Open the view, if it is not already open).



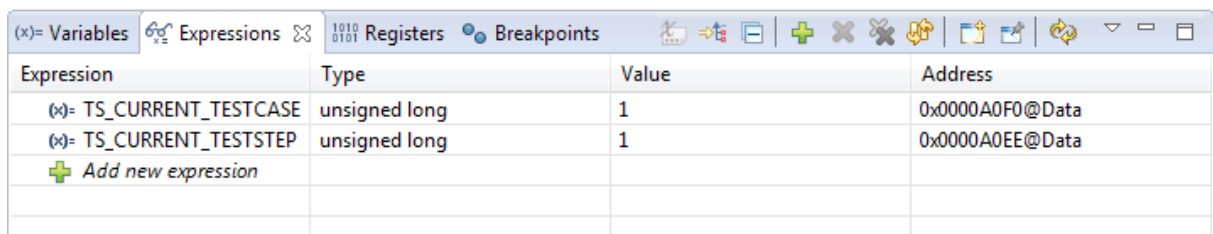
Paste the name of your test object into the dialog's **Location** field and click **OK**.



Now press the **Resume** button within CCS and the test execution will stop at your test object. The test data of the first test case is now available within the input variables of your test object. You may now step through and debug your test object.



The global variables `TS_CURRENT_TESTCASE` and `TS_CURRENT_TESTSTEP` contain the current test case and current test step number. Feel free to add them to the **Expressions** view.

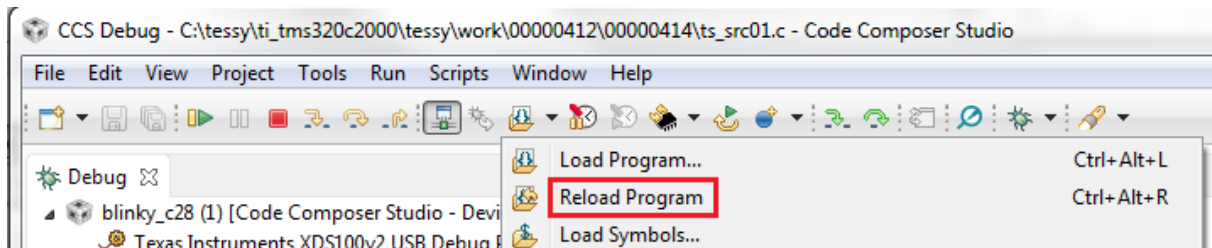


2.4 Troubleshooting When Debugging

2.4.1 Reloading the test application

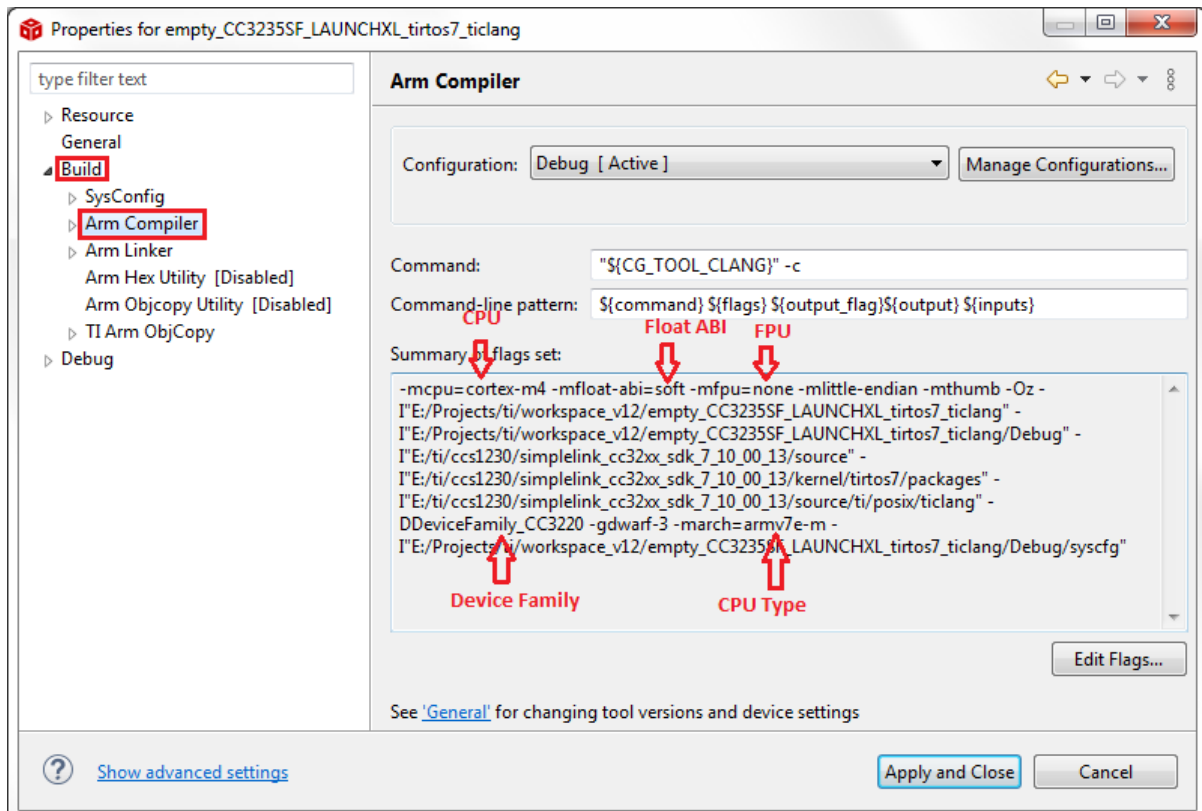
After changing your test data, you need to do the following steps:

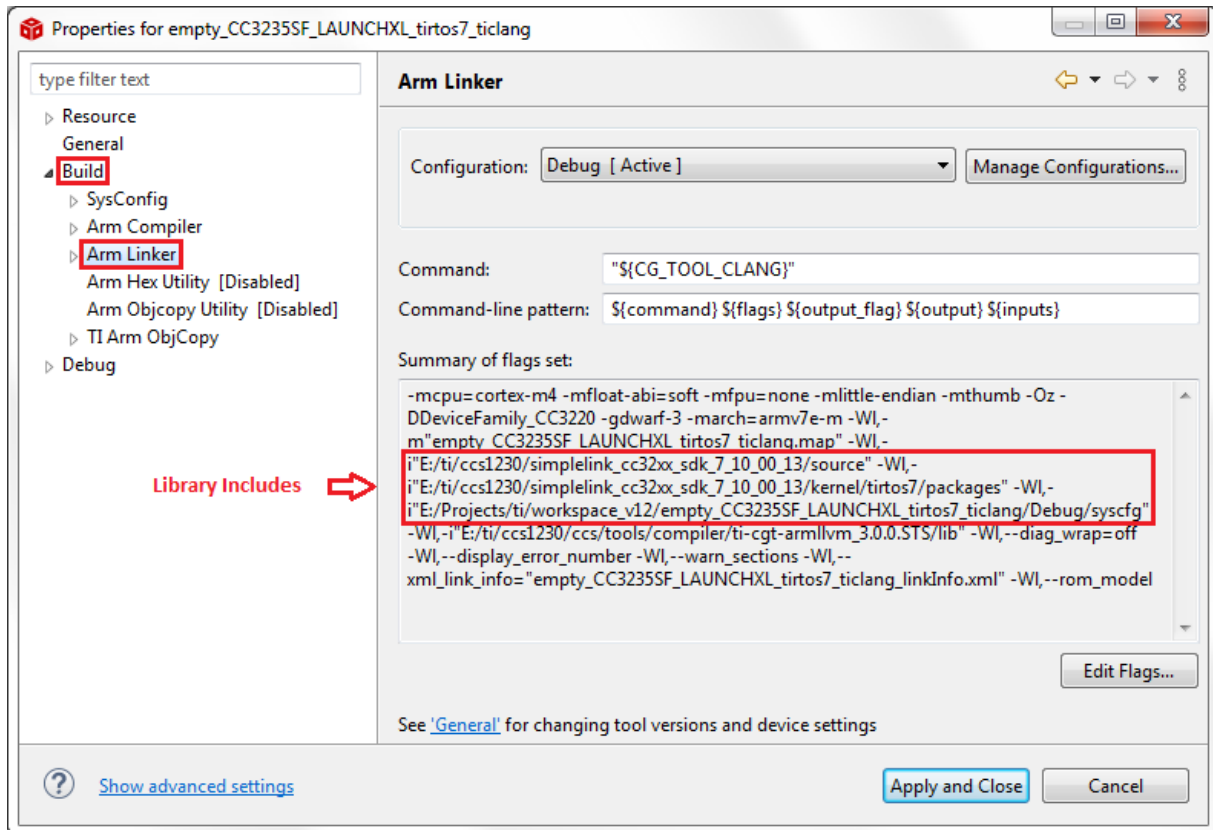
- Execute your tests from within TESSY again.
- TESSY will then rebuild your target binary automatically
- You may now reload the target binary from within CCS by clicking the **Load** toolbar menu button and select **Reload Program** in order to debug your application within CCS.



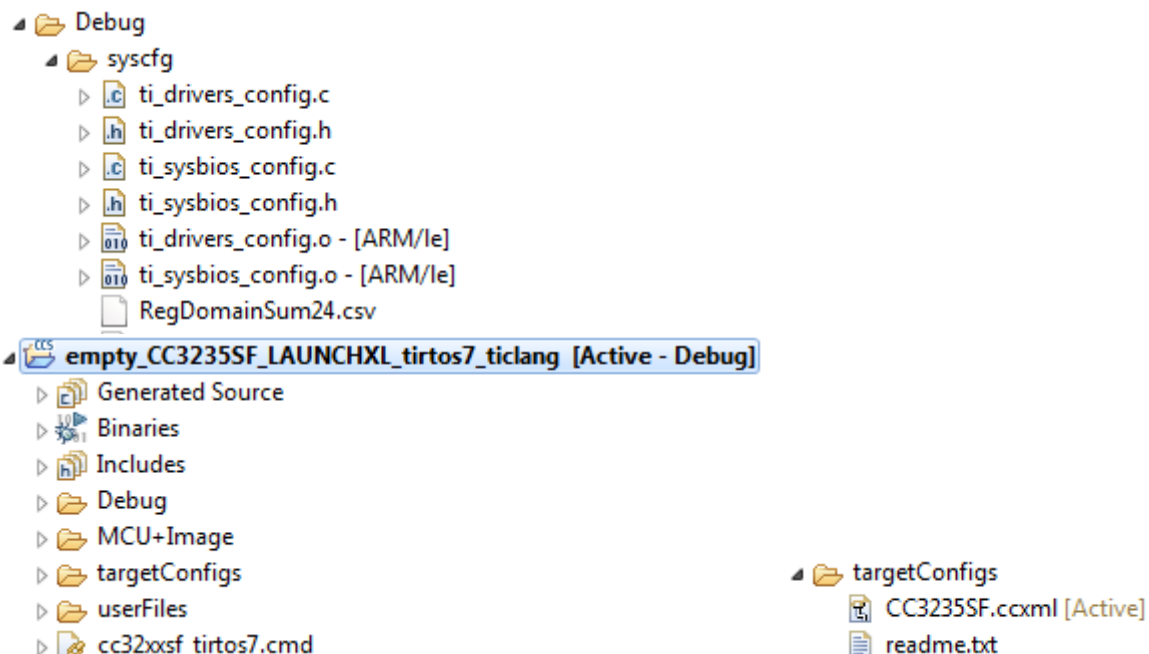
3 Using TI ARM CLANG Compiler

This chapter describes the TEE setup for the TI ARM CLANG compiler based on Code Composer Studio 12. The following CCS 12 snapshots show where to find the proper values for the respective TEE attributes. Displayed is the Code Composer Studio project's **Properties** dialog. Please, fill in all TEE attributes accordingly.



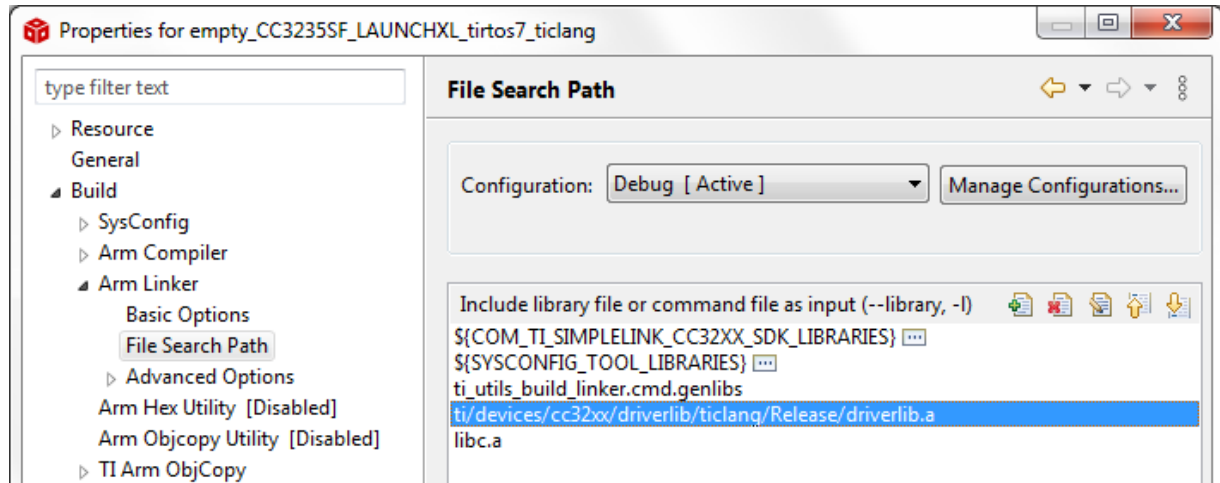


Furthermore, startup code (.o), a linker file (.cmd), and the target configuration file (.ccxml) are needed, which can be found and copied from the same project's **Project Explorer**.



Create a new folder of your choice inside your TESSY project folder. Copy the startup code into that folder. Put the absolute path name to this folder into TEE attribute **InitObjDir**. Copy the linker file and the target configuration file into the config folder of your TESSY project. Let TEE attribute **Linker File** point to the linker file and let TEE attribute **Target Configuration** point to the target configuration file.

Finally, the driver library file has to be found and its absolute path to be put into TEE attribute **Driver Library**. The library is probably found inside your device driver package installation folder.



4 CCSv2/CCSv3

The integration between TESSY and the Code Composer Studio (CCS) is carried out using the GEL language of CCSv2 and CCSv3. Since version 4 CCS provides another debugger API which is described within the previous chapter.

The test data is transferred using data files. To start the communication on startup, the default GEL init file need to be replaced by TESSY before the test execution starts and restored afterwards.

4.1 Files and Settings

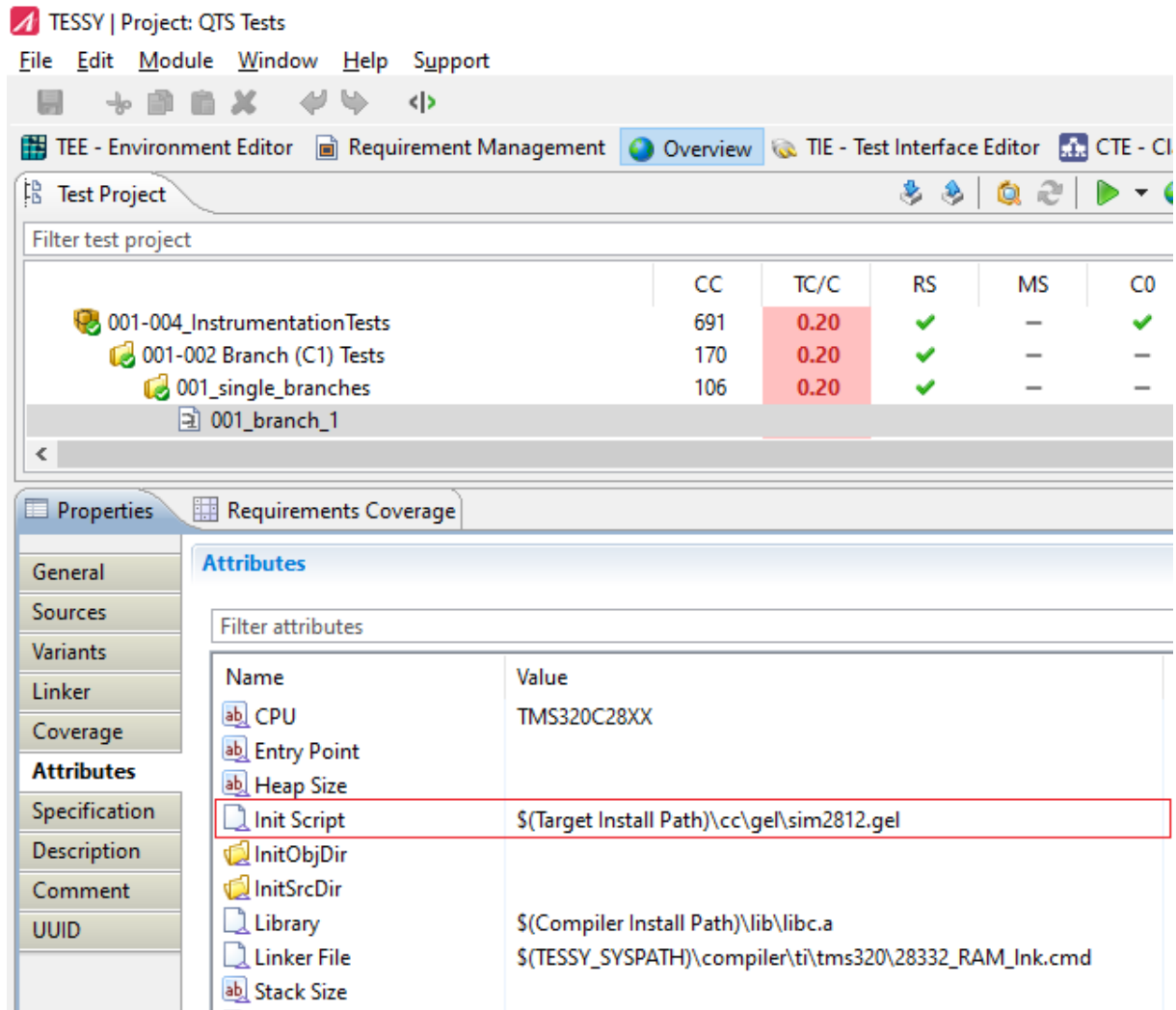
There are two important settings which have to be verified:

- the first is the CPU to be used
- the second is the GEL startup file to be replaced by TESSY.

These settings may be changed within the respective section of the Environment Editor (TEE) or within the module properties.

4.1.1 Module attribute settings

The default module attribute **Init Script** refers to the standard GEL startup file for the CCS simulator. If you are using an emulator connected to CCS you will probably need to change this setting: Simply enter the name of the GEL startup file that you are actually using.



The screenshot shows the TESSY application window for 'Project: QTS Tests'. The 'Requirements Coverage' tab is active, displaying a table of test project coverage:

Test Project	CC	TC/C	RS	MS	CO
001-004_InstrumentationTests	691	0.20	✓	–	✓
001-002 Branch (C1) Tests	170	0.20	✓	–	–
001_single_branches	106	0.20	✓	–	–
001_branch_1					

The 'Attributes' tab is also visible, showing the 'Init Script' attribute highlighted in red. Its value is: `$(Target Install Path)\cc\gel\sim2812.gel`.

Important note: The standard GEL startup file will be moved by TESSY and be replaced by a TESSY generated file. Please save your original file before running the test with TESSY.

The file pointed to by **Init Script** will be temporarily replaced by a TESSY generated file (for the time of test execution) based on the following template file:

```
...\sys\targets\ccs\master_gel.tpl
```

If specific settings are required within your GEL startup file (e.g. to setup the emulator memory map), you need to modify this template file. Before each test run, the template file will be expanded with the current module settings and be copied to the Init Script.

4.1.2 Project file

The CCS project file will also be copied from

```
...\sys\targets\ccs\ti_tms320.pj1
```

to the test area directory (e.g. C:\TESSY). The **CPU** attribute will automatically be inscribed into this file. You may change the template file if required.

4.1.3 Adapting the GEL startup file

The GEL startup file controls the initialization of the emulator and the target (e.g. setting the memory map or disabling the watchdog). Those functions should be copied into the GEL master template as described above.

Every command to be executed **before** reset of the target shall be added within the **StartUp()** function:

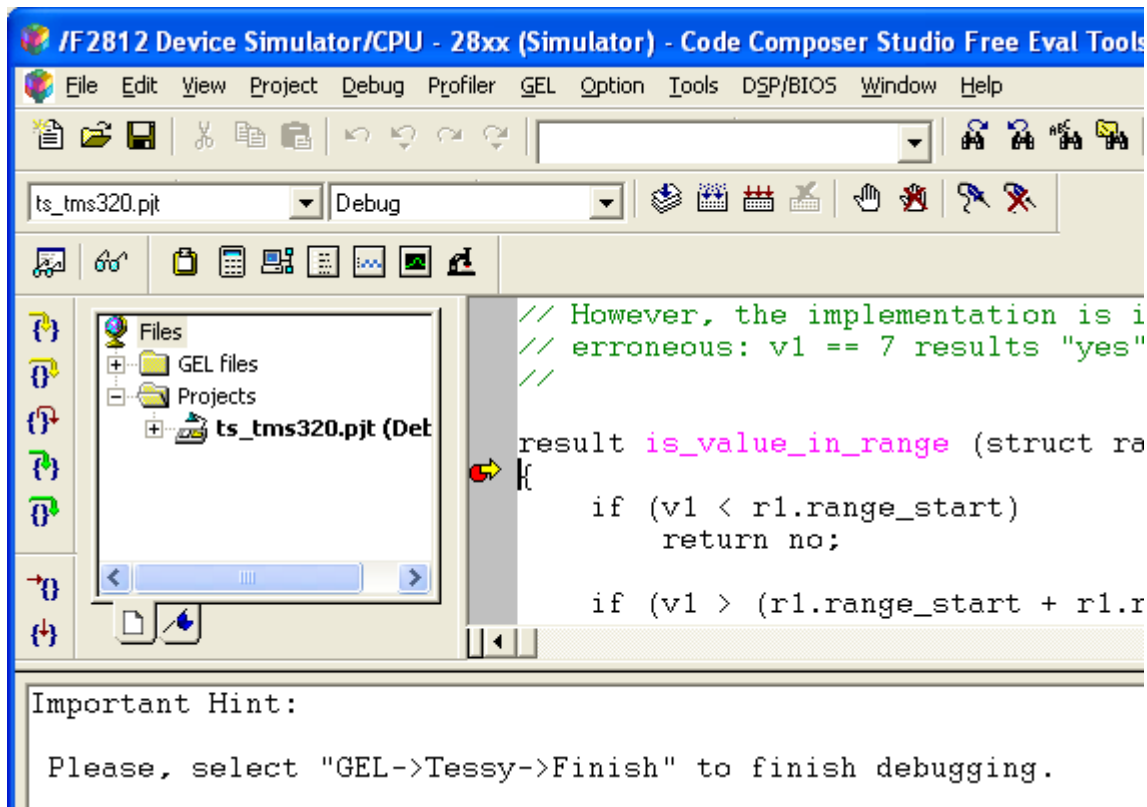
```
// Start up.  
StartUp()  
{  
    // Add your own specific  
    // initializations, if required ...  
    //  
    F2812_Memory_Map();  
    Enable_DFT();  
  
    // Run.  
    run();  
}
```

All settings, that need to be executed **after** resetting the target, shall be added to the **run()** function (e.g. disabling the watchdog):

```
// Initialize and go to probe points.  
run()  
{  
    // Initialize.  
    GEL_Reset();  
    GEL_Restart();  
    Disable_WD();  
  
    // Load project.
```

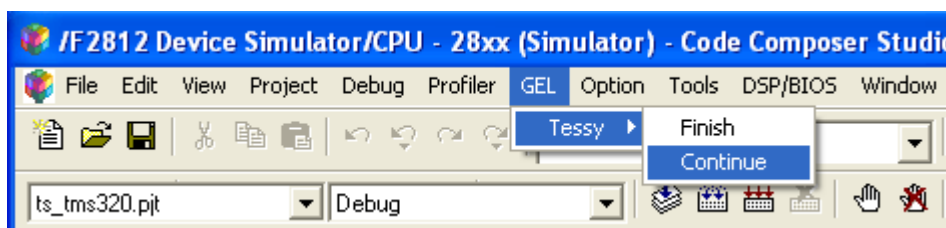
4.2 Running the Test

Code Composer Studio will be started automatically by TESSY upon test execution. The GEL startup file will be executed, which controls the connection to TESSY. If you set a breakpoint at the test object, you will see the following screen within CCS2:



You may now step through the test object or continue debugging using the **GEL->TESSY** menu:

- **Finish** executes the test until the end and exits CCS
- **Continue** runs the test up to the next test step



Note: If you do a new installation of TESSY and CCS, after the installation process has finished, you need to start CCS at least once manually before starting automatically from TESSY.

4.3 Compiler Specific Settings

With CCS3 there is a compiler switch available which allows changing the sizes of **double** types:

```
-f no-short-double
```

When using this option, you need to adapt the TESSY type settings within the `typetable.xml` file. The float type needs to be changed to 64 bit in this case (the figure below shows the default setting for use without this option):

```
2716 <compiler id="TS_COMPILER_TI_TMS470" double_size="64" enum_&
2717 <typemap sign="plain">
2718 <sizemap size="none">
2719 <type name="none" value="IDB_TYPE_S32"/>
2720 <type name="void" value="IDB_TYPE_VOID"/>
2721 <type name="char" value="IDB_TYPE_CHAR"/>
2722 <type name="int" value="IDB_TYPE_S32"/>
2723 <type name="float" value="IDB_TYPE_FLOAT32"/>
2724 <type name="double" value="IDB_TYPE_FLOAT64"/>
2725 <type name="wchar" value="IDB_TYPE_WCHAR"/>
2726 </sizemap>
```