VECTOR

VectorCAST RSP TASKING with Tricore Aurix

Version 1.0

2023-03-13

Application Note AN-ACT-1-106

Author Derek Opitz

Restrictions Public Document

Abstract Information on the VectorCAST RSP TASKING with Tricore Aurix

Table of Contents

1 Overview

VectorCAST Runtime Support Package (RSP) is an extension of the VectorCAST product. It provides an interface layer that allows the usage of VectorCAST testing techniques and methods on an embedded target processor. The VectorCAST RSP will always run on the host platform (the same platform as the compiler). Only the VectorCAST generated test harness will be downloaded to, and executed on, the embedded target.

A VectorCAST RSP is always customized to a particular Target CPU, Cross Compiler, and Run-Time environment (or kernel). As such, the information presented here contains sections for this specific RSP.

This application note provides details on configuring VectorCAST to run tests on a Tricore Aurix processor using the TASKING compiler. The target can be a live board, or a simulated board using the TASKING simulator.

VectorCAST uses the TASKING compiler and linker to build the VectorCAST test harness into an executable image that can be loaded onto the Tricore Aurix processor or simulator. VectorCAST then uses the TASKING script debugger to load the executable image onto the Tricore Aurix processor or simulator and run the test and collect the test results.

VectorCAST provides off the shelf TASKING RSPs for the Tricore Aurix architecture on the TASKING simulator. If the user needs support for a different architecture, need help configuring VectorCAST for their processor or debugger, or run into any issues configuring VectorCAST, please contact support@vector.com.

2 Setup

2.1 Use Case: Selecting the Proper Configuration

The first thing to do when configuring TASKING RSP is to select the correct Tricore Aurix CPU as the target as seen in Figure 1.

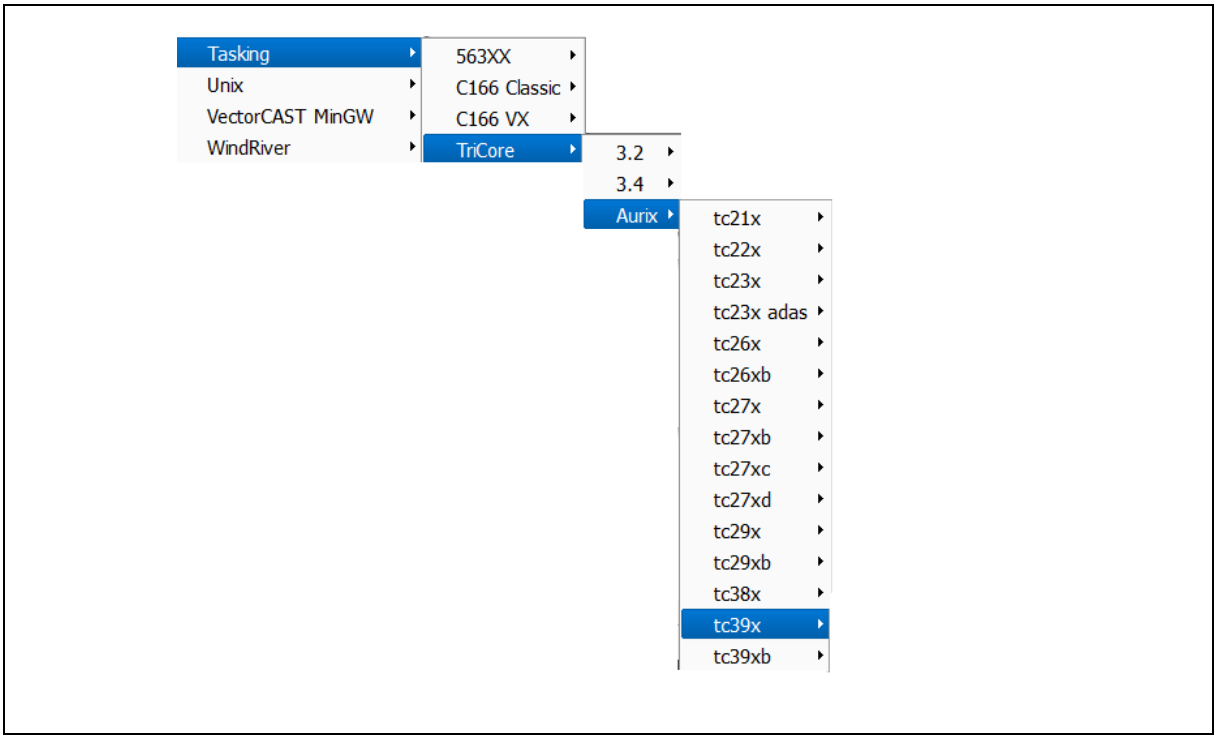

Figure 1 – Selecting the TASKING Configuration

The second step is to select the simulator to be used for test execution. Note: This App Note only covers the Tasking simulator. If you want to use Trace32 or VLAB please contact support@vector.com for further assistance.

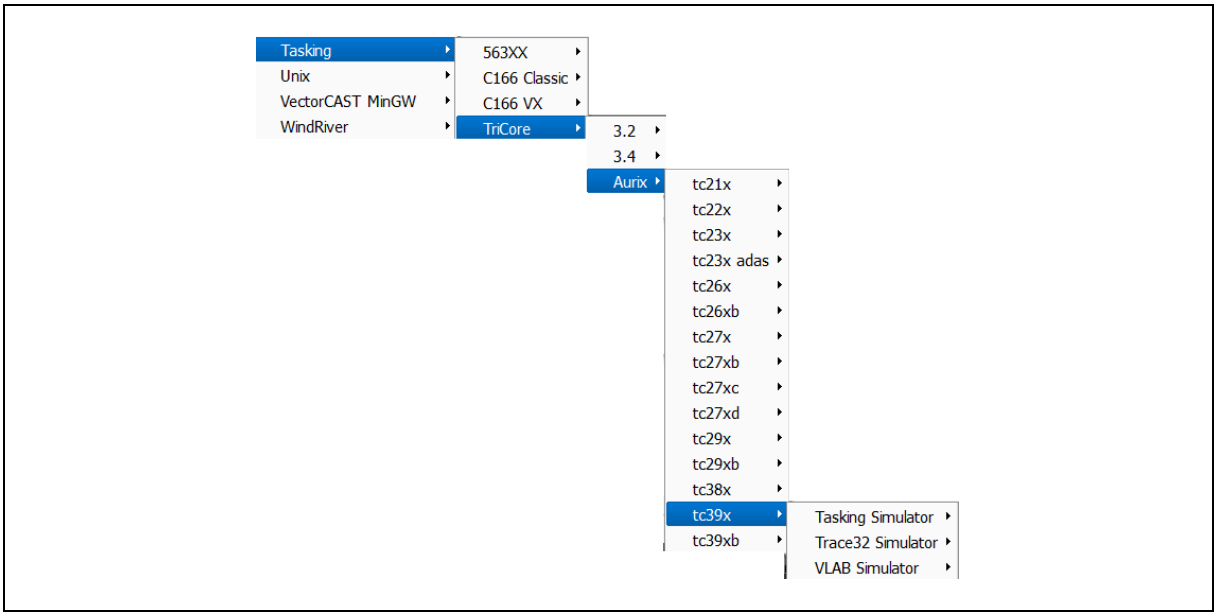

Figure 2 – Selecting the Simulator

The last step is to select the type of code to test. Choose C for only testing C code, choose C++ for either C++ or a combination of C and C++.

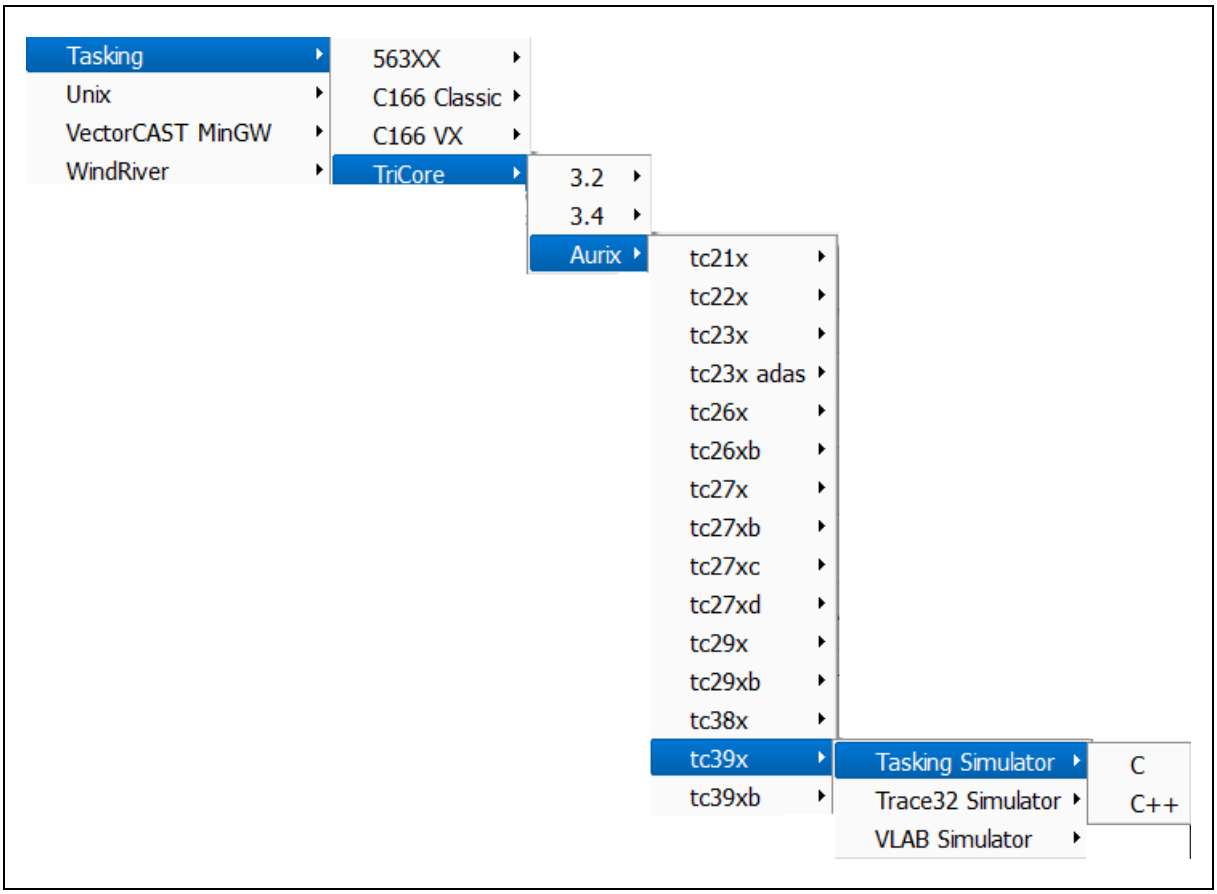

Figure 3 – Selecting the Coding Configuration

Once selected either C or any C++ version, the configuration will show up in the project and the compiler node will be populated with the proper data, seen in Figure 4.

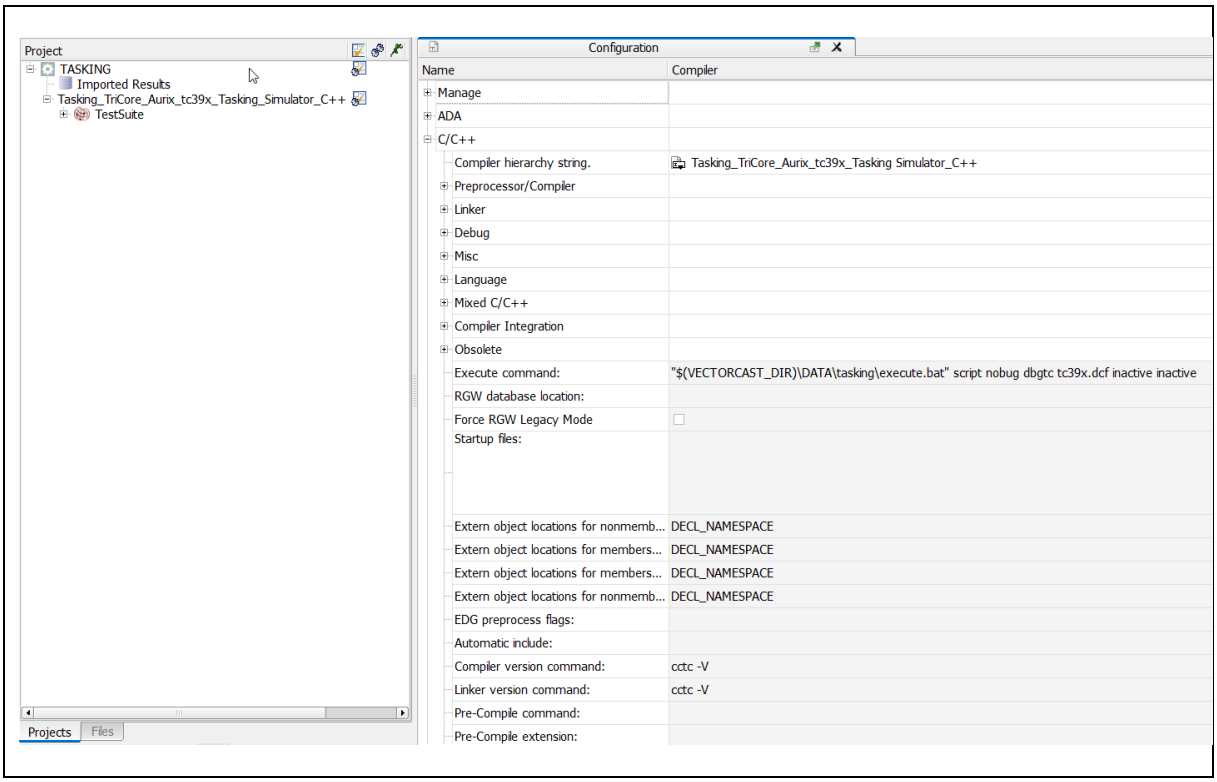

Figure 4 – Project Configuration

2.2 Use Case: Environment Setup

VectorCAST needs to know where the TASKING tools have been installed. VectorCAST determines this using the environment variable VCAST_TASKING_TRICORE_INSTALL_DIR. Additionally, the path to the TASKING compiler and the TASKING script debugger needs to be on the path environment variable.

VectorCAST recommends putting the setting VCAST_TASKING_TRICORE_INSTALL_DIR and the path into a startup script called **setup_VectorCAST.bat** or **setup_VectorCAST.sh** depending on if using Windows or Linux.

setup_VectorCAST.bat script will end up looking something like this:

```
set VCAST_TASKING_TRICORE_INSTALL_BASE=C:\TASKING\TriCore_v6.3r1
set
path=%VCAST_TASKING_TRICORE_INSTALL_BASE%\ctc\bin;%VCAST_TASKING_TRICORE
_INSTALL_BASE%\bin;%path%
```

The first line of **setup_VectorCAST.bat** sets the environment variable VCAST_TASKING_TRICORE_INSTALL_DIR, the second line puts the path to the compiler and the

path to the TASKING script debugger on the environment path so that VectorCAST can call the compiler and the debugger.

The **setup_VectorCAST.bat** script will need to be called before execution of any VectorCAST CLI commands (such as `%VECTORCAST_DIR%\manage` or `%VECTORCAST_DIR%\clicast`).

The **setup_VectorCAST.bat** script will also need to be called before launching the VectorCAST GUI. VectorCAST recommends creating another script called **start_VectorCAST.bat** that will call **setup_VectorCAST.bat** and then launch the VectorCAST GUI.

start_VectorCAST.bat will look like this:

```
call %~dp0setup_VectorCAST.bat
start %VECTORCAST_DIR%\vcastqt
```

The `%~dp0` is a windows batch syntax that will expand the directory path so that **`setup_VectorCAST.bat`** will be called from the same directory that **start_VectorCAST.bat** exists in. This allows for a shortcut to **start_VectorCAST.bat** to be used and if **setup_VectorCAST.bat** is in the same directory as **start_VectorCAST.bat** then the shortcut will work.

Use **start_VectorCAST.bat** to launch VectorCAST to be able to build or execute unit test environments.

2.3 Use Case: Building Environments

Following the setup instructions in this App Note should allow the build of VectorCAST environments for any of the supported Aurix processors. If encountering trouble with any of the supported Aurix processors, please contact support@vector.com.

2.4 Use Case: Testcase Execution

Following the setup instructions in this App Note should allow the execution of testcases on the TASKING simulator without any further modifications. If executing testcases on a live target, then modifications are required (details below).

VectorCAST uses the Execute command in the VectorCAST options to execute tests. For the TASKING compiler, VectorCAST uses a batch script to call dbgtc, which is the TASKING script debugger. The TASKING script debugger uses a configuration file to determine the processor and debugger (either hardware debugger or simulator).

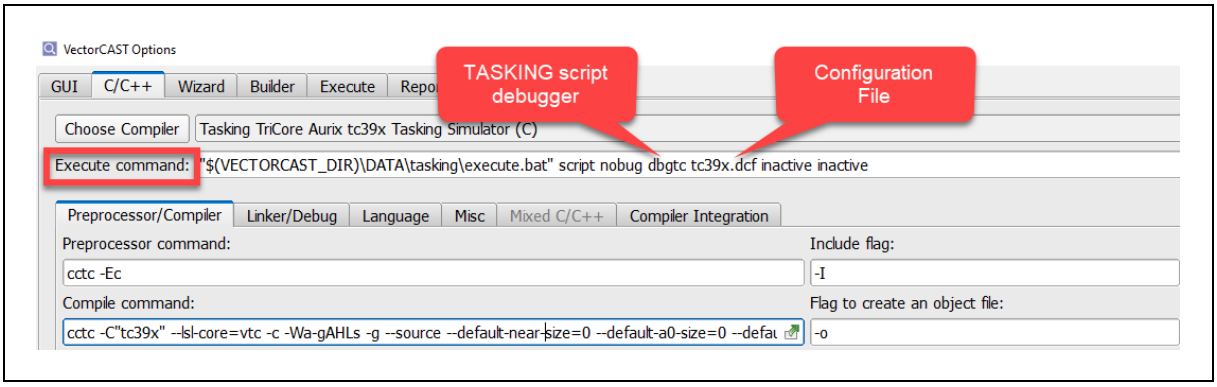

Figure 5 – Testcase Execution

For the TASKING simulators, TASKING script debugger configuration files have been provided for most Aurix processors. If running on a live board then a TASKING script debugger configuration file must be created. If further assistance is needed in creating a TASKING script debugger configuration file, then contact support@vector.com for instructions and assistance.

2.5 Use Case: Debugging

Debugging VectorCAST testcases is done using the TASKING Eclipse IDE.

The following steps will describe the process for debugging a testcase from the TASKING Eclipse IDE.

When right clicking on a testcase and select **'Execute with Debug'** VectorCAST will launch the TASKING Eclipse IDE. The first thing that is required to debug the VectorCAST harness is to create a debug configuration. Before a debug configuration can be created, the TASKING Eclipse IDE requires a project to be configured. The following steps will describe the process for creating a project.

Select **File->New->TASKING TriCore C/C++** Project and create a Hello World C or C++ Project and select **Next**.

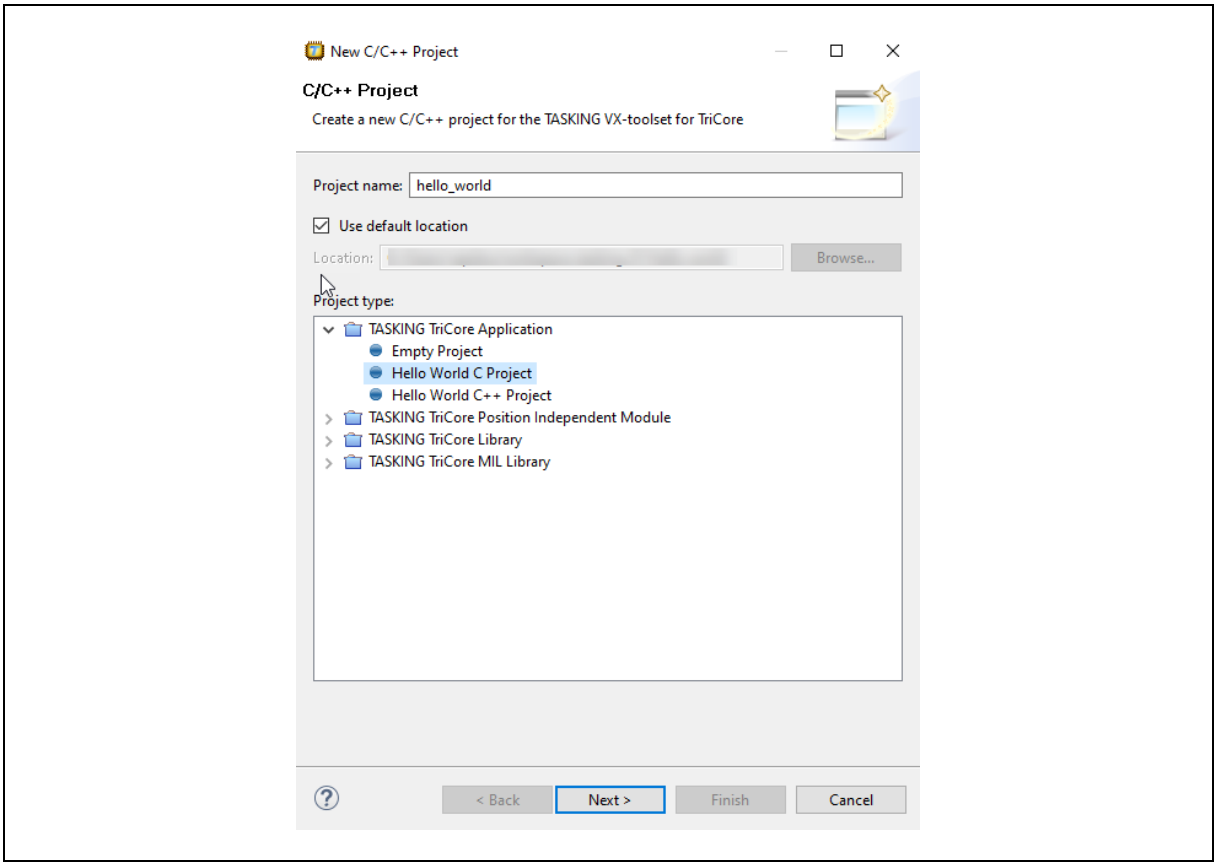

Figure 6 – New Hello World Project

On the next screen, select the correct chip for the project and select **Next**.

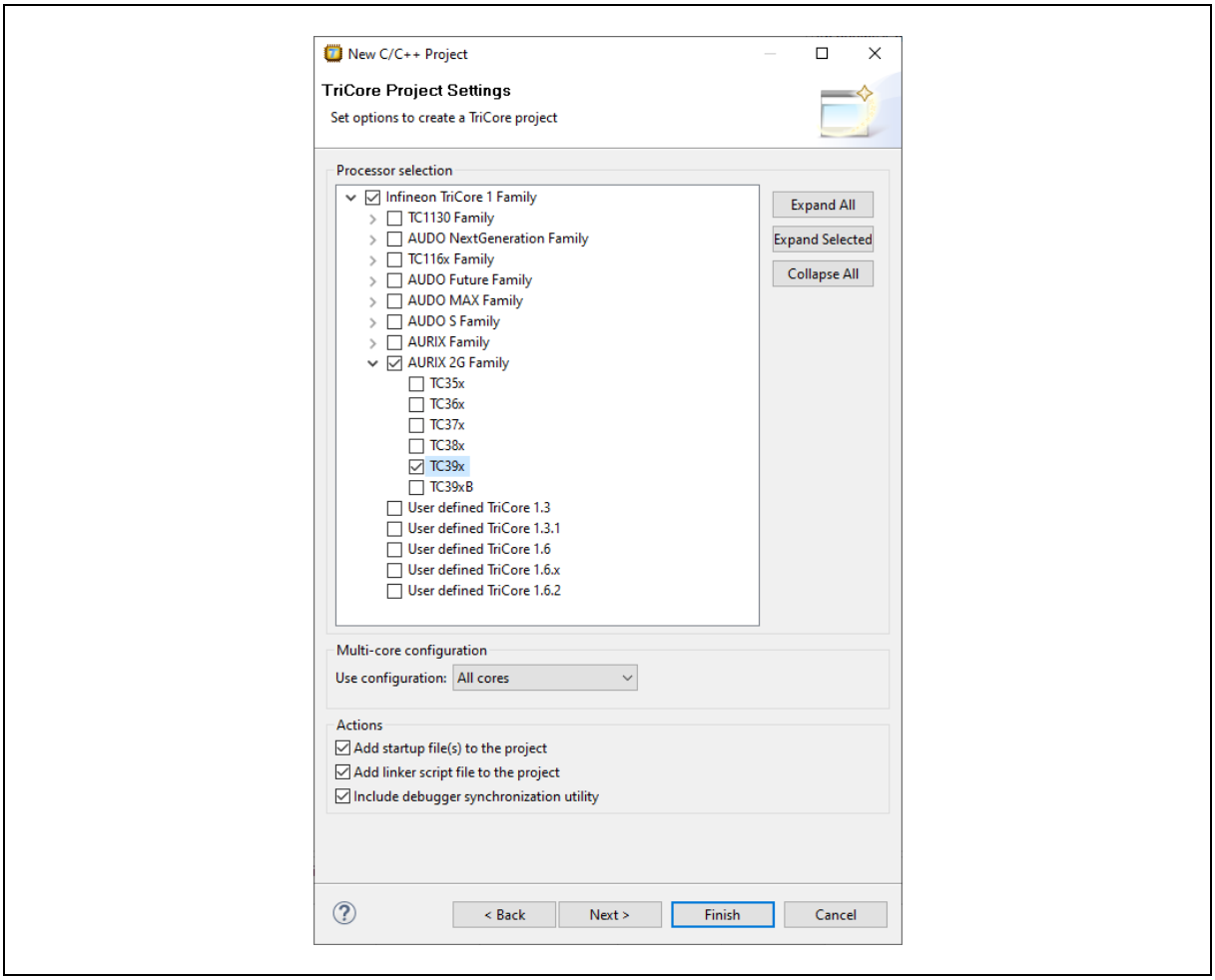

Figure 7 – Processor Selection

On the next screen, if running on the simulator then select the simulator Target. If running on a target CPU then select the correct target and verify the connection settings are correct. Then select **Finish**.

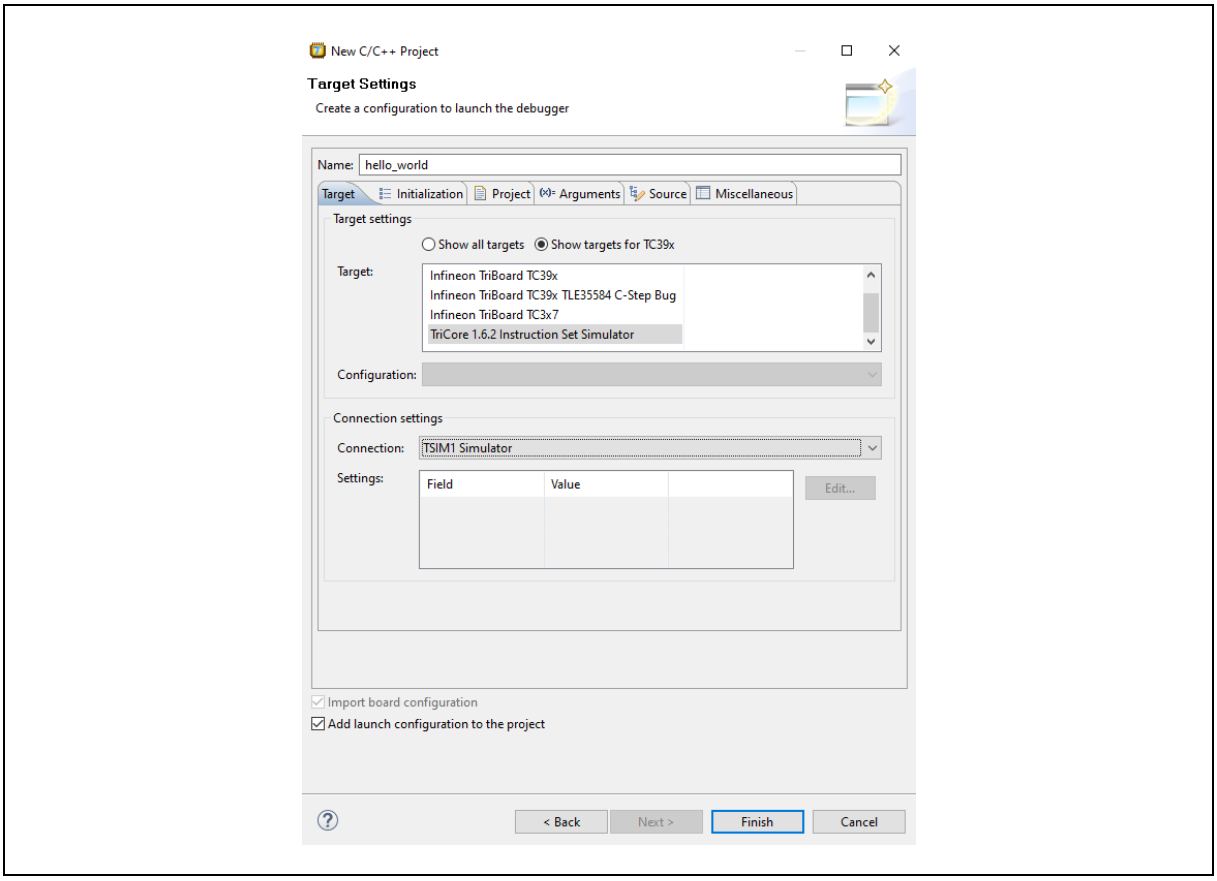

Figure 8 – Processor Selection

At this point, a new hello world project has been created, as well as a debug configuration for this hello world.

The next step is to create a debug configuration for the VectorCAST harness. To create a debug configuration for the VectorCAST harness, select **Debug->Debug Configurations…** from the top-level Debug menu. Next, duplicate the hello world configuration by right clicking on the hello world configuration and choosing **duplicate**. Next, click on the new configuration, change the name to vcast_debug, and select **Apply**.

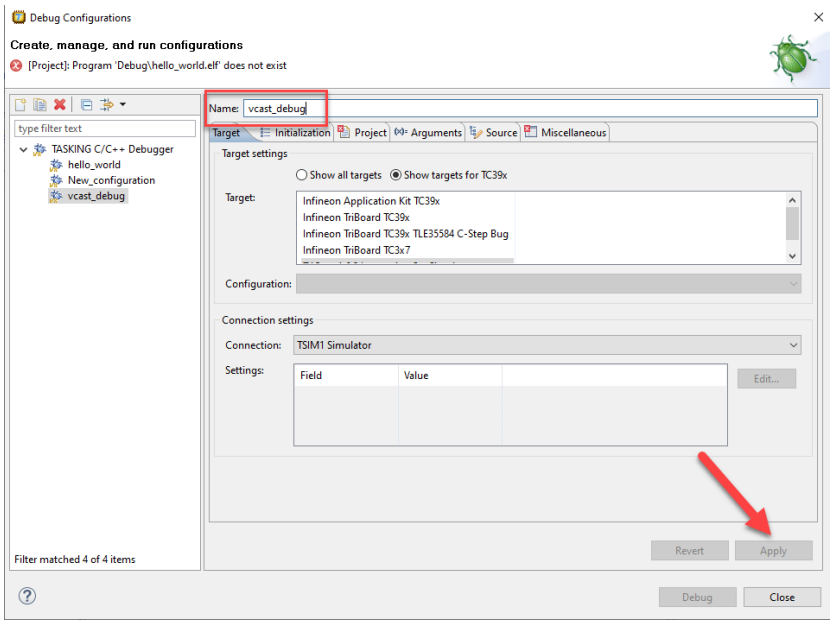

Figure 9 – Processor Selection

Next, on the **Initialization** tab, verify that the top five boxes are checked.

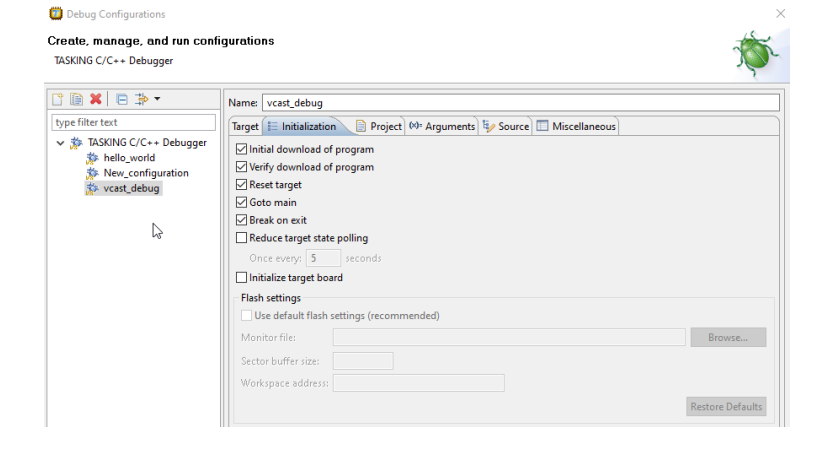

Figure 10 – Test Harness

Next, on the Project tab change the name of the executable file to ${env_var:VCV_ENVIRONMENT_DIR}\${env_var:VCV_EXECUTABLE_NAME}. This will make the debug configuration generic so that any environment being debugged from will select the correct file for debugging.

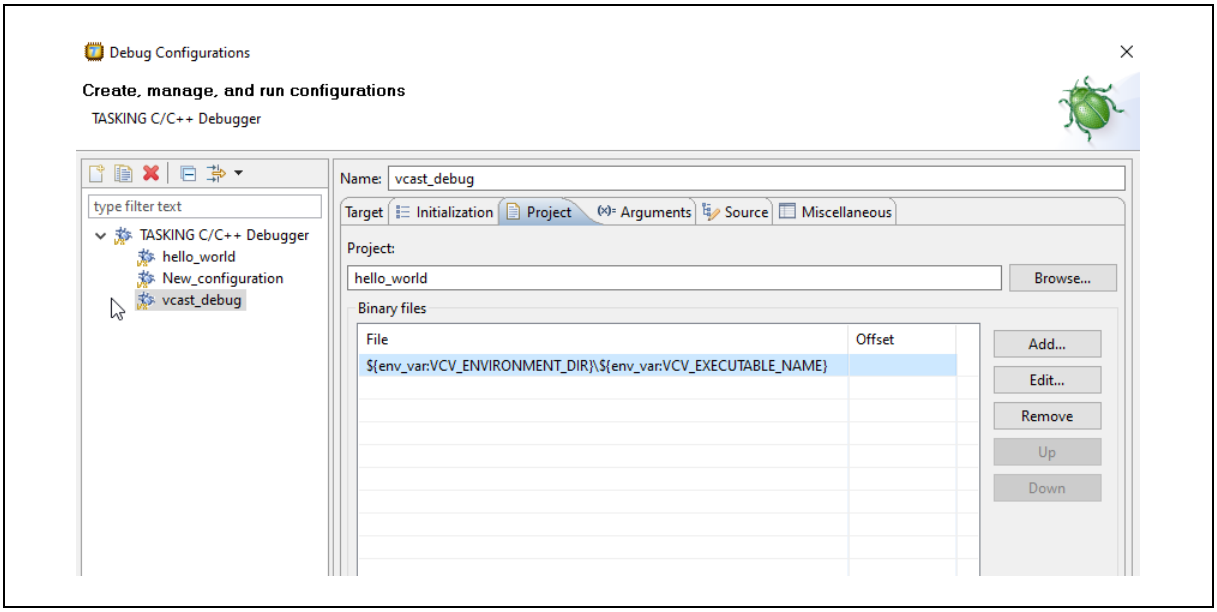

Figure 11 – Test Harness

Next, on the **Miscellaneous** tab, delete the path in the FSS root directory. This will make the FSS root directory the directory that Eclipse was launched from, which is the VectorCAST environment directory.

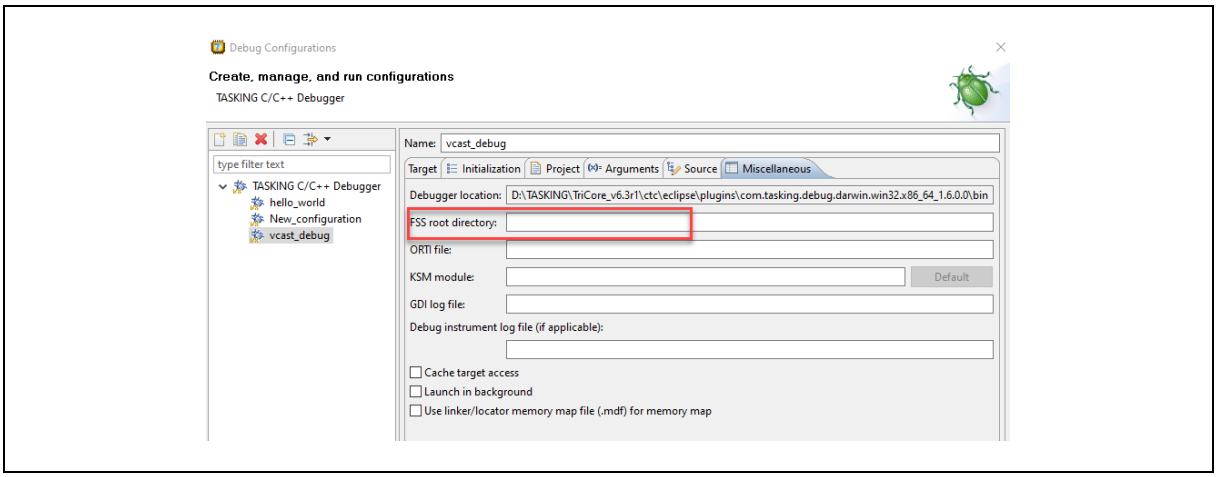

Figure 12 – Test Harness

Select **Apply** and then **Close**. At this point, VectorCAST testcases are ready to be debugged. Using the pulldown to the right of the Debug icon, select vcast_debug.

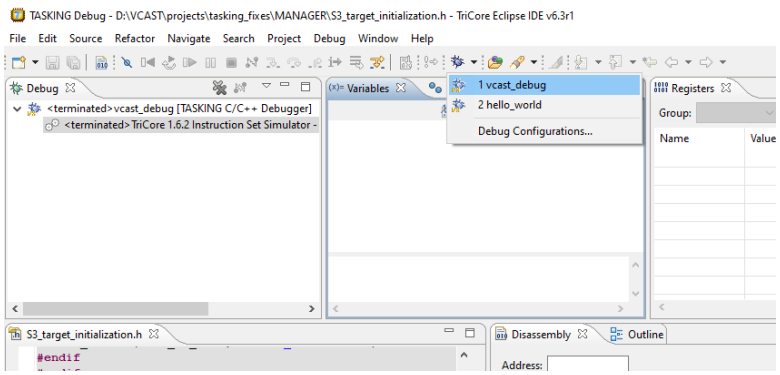

Figure 13 – Test Harness

Execution should stop at main of the VectorCAST harness and ready to debug.

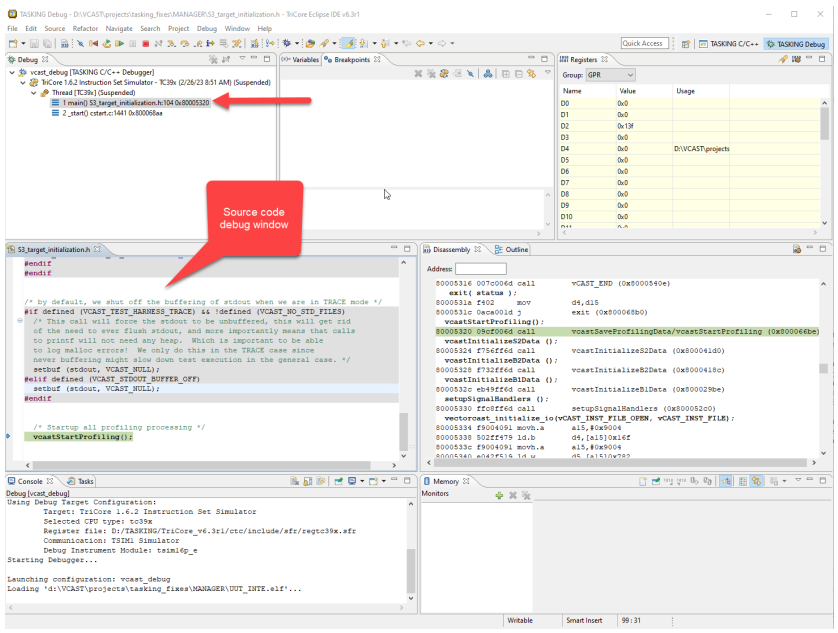

Figure 14 – Test Harness

VectorCAST recommends setting a breakpoint on the function under test if debugging customer specific code. Select the **Add TASKING** breakpoint in the Breakpoints window.

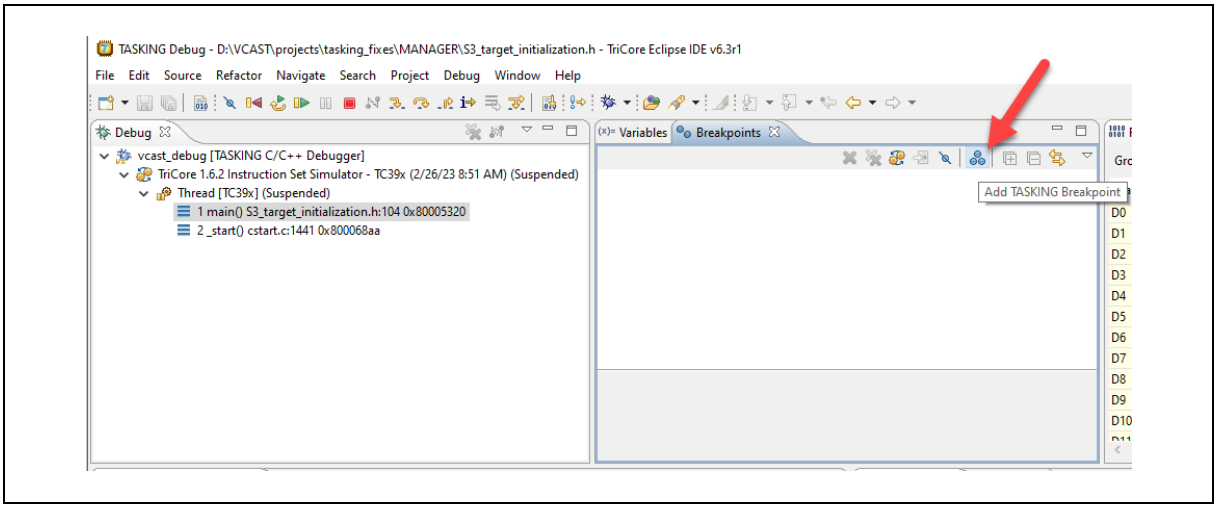

Figure 15 – Test Harness

Next, go the **Function** tab and use the pulldown to select the function under test.

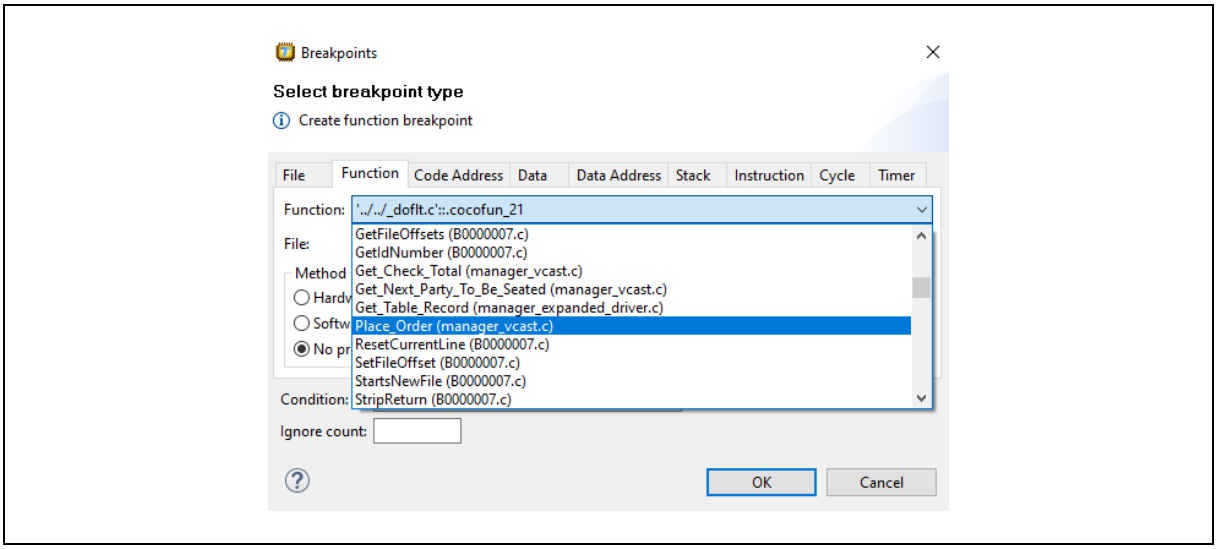

Figure 16 – Test Harness

Next, hit the **Resume** button on the toolbar and execution should stop on the function under test and allows for debugging. The VectorCAST harness output data will be shown in the FSS tab in the bottom right-hand corner.

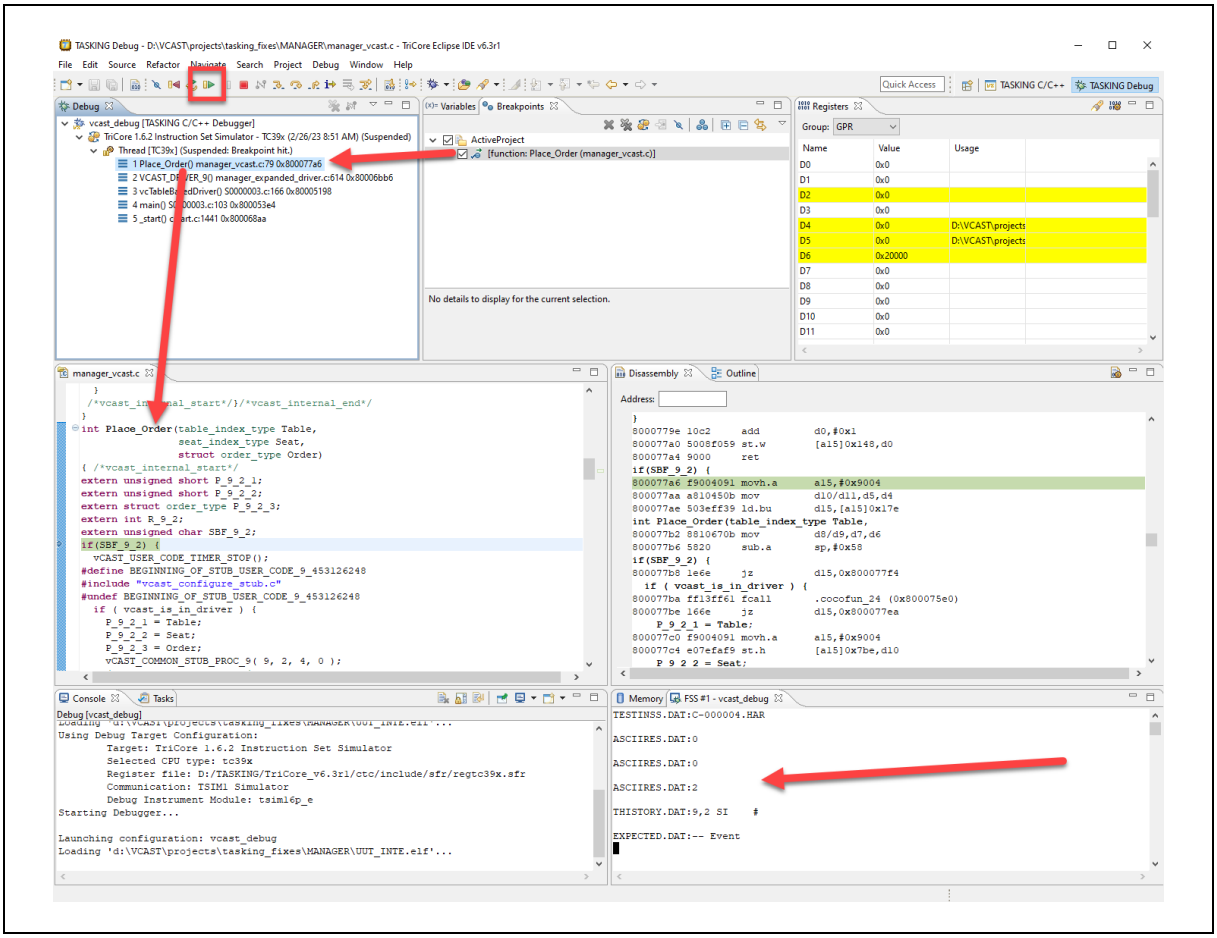

Figure 17 – Test Harness

3 Additional Resources

https://www.tasking.com/products/tricore-vx-software-development-tools

4 Trademarks

All mentioned names are either registered or unregistered trademarks of their respective owners.

5 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.