The Test Configurations that are provided for testing Code Composer projects are preconfigured to handle most typical projects configurations. However, it may be necessary to modify the project configuration to enable runtime testing. Common modifications include:

- Due to the testing framework's additional resource consumption, you may need to increase the default size of the heap and stack.
- Due to the testing framework's additional resource consumption, you may need to change the program layout in the memory. In some situations, the test executable linked with the C++test runtime library does not fit into the internal signal processor memory and needs to be located into external RAM. This may require changes in the project's linker script.
- It has been observed that unit test execution may be significantly slower than the original program's execution. There are few factors that contribute to this: C++test's instrumentation overhead; the use of the TI runtime support library for channeling file i/o operations to the host platform (for storing test results), and using slower external memory for program execution. Performance can be improved by limiting instrumentation features; you can configure limited instrumentation in **Test Configuration> Execution tab> Instrumentation mode**. The best results can be achieved with a "no instrumentation" configuration (running only regression unit tests without coverage or stack trace reporting).
- The built-in Test Configurations Builtin> Embedded Systems> Texas Instruments> Run TI CCS v4+ Tests and Run TI CCS v4+ Application with Memory Monitoring both contain a built-in step for preparing the C++test runtime library. When this step is present, you do not need to manually build the C++test runtime library and add it to the linker flags. If you want to prevent the automatic build of the runtime library, go to the Test Configuration's **Execution > General** tab, click the **Edit** button next to **Test Execution flow**, then remove `<BuildRuntimeLibStep />` from the xml document describing the test flow execution.

To facilitate testing with the Code Composer environment, a simple connecting JavaScript is provided: `<C++test install directory>/engine/bin/CCS4xConnector.js`. This program assists with deploying and running the test binary on the target or simulator.

The "Run TI CCS v4+ Tests" and "Run TI CCS v4+ Application with Memory Monitoring" Test Configurations use the active target configuration from the Code Composer project to execute the prepared tests. If there is no "Active" target configuration selected in the project, then "Default" will be used (if there is no configuration marked as "Active" or "Default", then the first available target configuration file will be selected). If you want to change this behavior, you can specify the target configuration file manually, as a value of the test execution flow property. It can be accessed in Test Configuration's **Execution> General** tab (in the **Execution details** field). The target configuration file should be specified with a full path.

The CCS utility is based on the `loadti` example script that is shipped with Code Composer Studio distribution. The script is located in `<Code Composer Install dir>/ccs4/scripting/examples/loadti` (for 4.x) or `<Code Composer Install dir>/ccsv5/ccs_base/scripting/examples/` (for 5.x). Refer to the example's documentation for details on advanced options.

Once started, the `CCS4xConnector.js` utility will try to connect to Debug Server, load the test binary, and execute it.

Unit Testing

Before you run unit tests on an existing Code Composer project, you might need to perform some of the customizations described in Runtime Testing.

To run generated unit tests:

1. Select the desired context for the unit testing.
2. Run the analysis session by selecting the "Builtin> Embedded Systems> Texas Instruments> Run TI CCS v4+ Tests" Test Configuration.

If you need to customize the builtin Test Configuration:

1. Duplicate the "Builtin> Embedded Systems> Texas Instruments> Run TI CCS v4+ Tests" Test Configuration.
2. Open the duplicated Test Configuration's **Execution> General** tab.
3. In the **Execution details** area, you can modify the following test execution flow properties:
    - **Target configuration file:** This property points to the debugger configuration file that you want to use for your testing process. By default the "Active" (or, if that is not present, the "Default") target configuration file will be used. You can specify any path to a valid target configuration file. Note that your environment needs to be correctly configured for using this target (including GEL files initialization). C++test will only attempt to connect to the target as specified—without introducing any extra initialization.
    - **Executable exit point symbol:** This property specifies the symbol that will be used for setting a breakpoint which will mark the end of test execution.

Debugging Test Cases

Use the standard means of executables debugging for TI Code Composer v4.

Use Eclipse Internal debugging mode for TI Code Composer v5.x and 6.0. See the following sections for details:

- Configuring Debugger Settings
- Debugging in Various Embedded Development Environments

Application Monitoring

Before you run application monitoring on an existing Code Composer project, you might need to perform some of the customizations described in Runtime Testing.

To run application monitoring:

1. Select the desired context for the monitoring.
2. Run the analysis session by selecting the "Builtin> Embedded Systems> Texas Instruments> Run TI CCS v4+ Application with Memory Monitoring" Test Configuration.

If you need to customize the builtin Test Configuration:

1. Duplicate the "Builtin> Embedded Systems> Texas Instruments> Run TI CCS v4+ Application with Memory Monitoring" Test Configuration.
2. Open the duplicated Test Configuration's **Execution> General** tab.
3. In the **Execution details** area, you can modify the following test execution flow properties:
    - **Target configuration file:** This property points to the debugger configuration file that you want to use for your testing process. By default the "Active" (or, if that is not present, the "Default") target configuration file will be used. You can specify any path to a valid target configuration file. Note that your environment needs to be correctly configured for using this target (including GEL files initialization). C++test will only attempt to connect to the target as specified—without introducing any extra initialization.
    - **Executable exit point symbol:** This property specifies the symbol that will be used for setting a breakpoint which will mark the end of test execution.