Runtime Testing with ARM RVDS

This topic explains how to configure runtime testing on code that is designed to be compiled/built using ARM RVDS.

In this section:

General Configuration

The Test Configurations provided to test RVDS projects are configured to automatically build the C++test runtime library. The communication channel preselected in the runtime library is based on the semihosted file I/O writes. If the support for semihosting is not available for a given platform, you need to customize the runtime library and implementation alternative communication channel. For more details see Working with the C++test Runtime Library

Customizing the RVDS Test Configurations

C++test provides the following built-in Test Configurations to analyze RVDS projects:

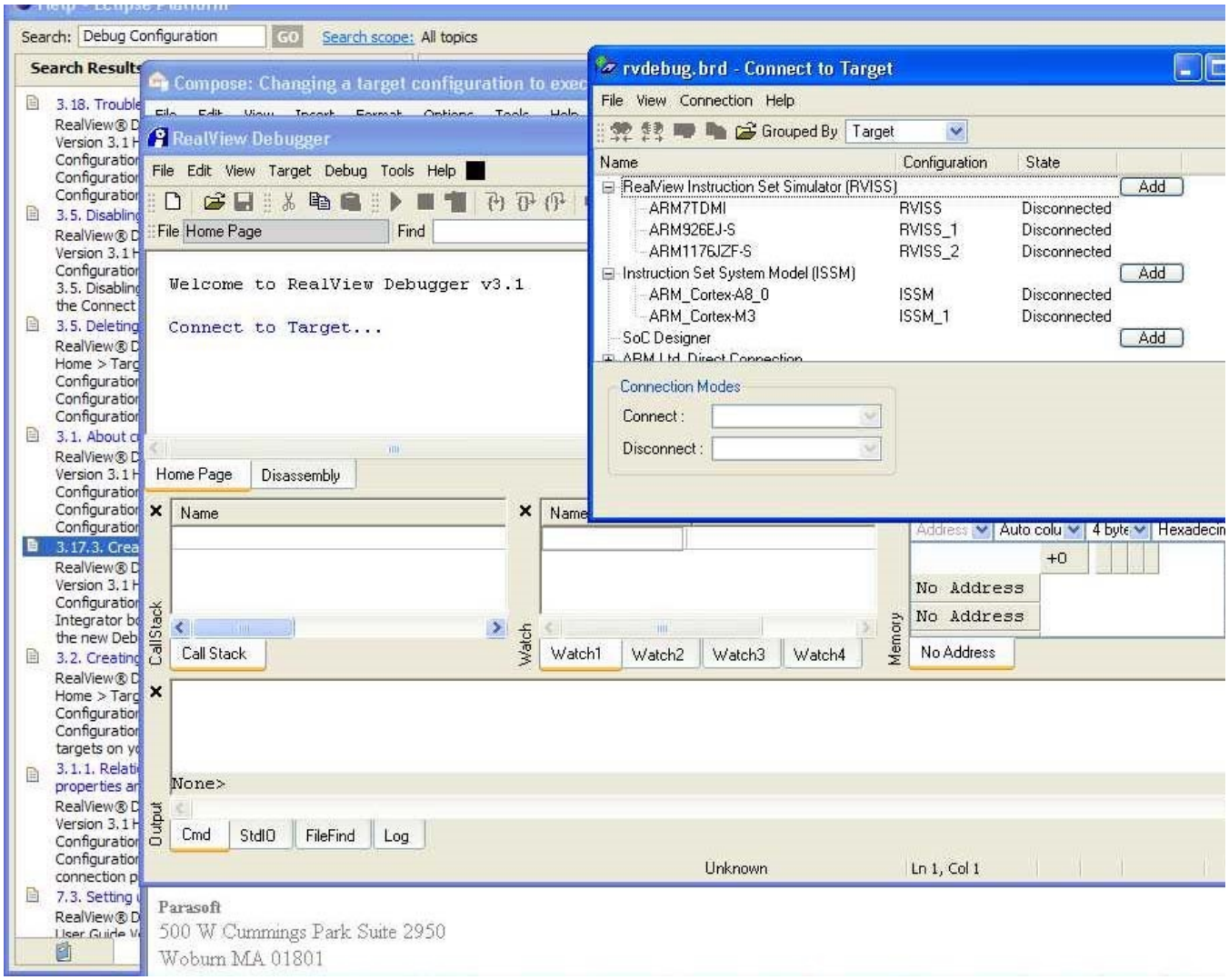
- Builtin> Embedded Systems> ARM> Run RVDS 3.x 4.x Application with Memory Monitoring
- Builtin> Embedded Systems> ARM> Run RVDS 3.x 4.x Tests

Both of these Test Configurations are designed to run the test executable on the target device or simulator. The process of loading and starting the test executable is handled via the RV Debugger, with the help of an automatically-generated script of the following form:

```
#This script generates ARM debugger's (rvdebug) run script WAIT ON
connect "$TARGET_DEVICE$"
load/r '$TEST_EXECUTABLE$;;$TEST_ARGUMENT$'
go
disconnect
quit y
```

The debugger script template file is available at `<C++test Install dir>\engine\etc\templates\for_recipes\arm_test.tja`.

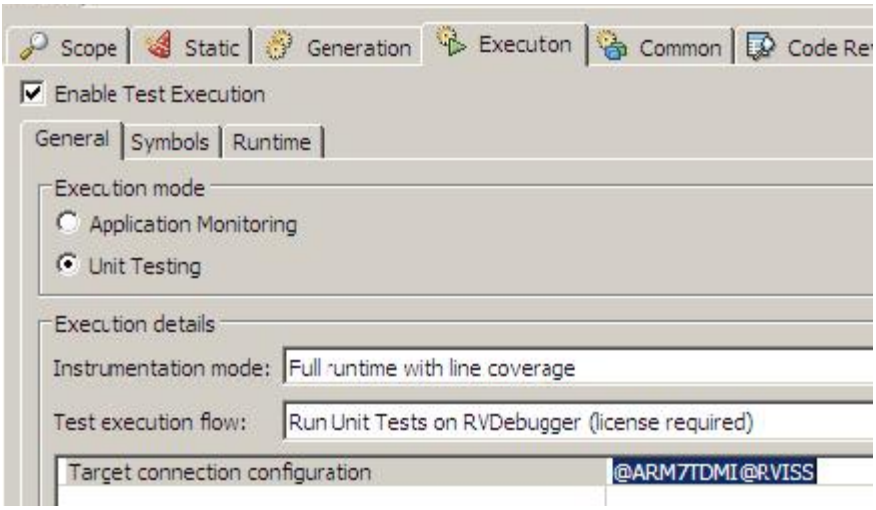
The specific RV Debugger Debug Configuration must be created before you can run a C++test Test Configuration with the RV Debugger. General instructions for creating a Debug Configuration in the RV Debugger are provided in the RV documentation. Once created, the debug configuration typically has a two-part name: the name of the target, and the name of the associated debug interface instance:



For example, one of the debug configurations would be encoded for the debugger as @ARM_Cortex-M3@ISSM_1.

Once such a debug configuration is configured, you perform the following steps in the C++test perspective of RVDS:

1. Choose **Parasoft> Test Configurations**.
2. Create a target-specific Test Configuration by right-clicking the built-in **Run RVDS 3.x, 4.x Tests** Test Configuration, then choosing **Duplicate**.
3. Rename the duplicated Test Configuration to reference your target.
4. Open that Test Configuration's **Execution> General** tab
5. In the **Execution details** area, modify the execution properties. Specifically, change the value of **Target connection configuration** to correspond to your target. For the above example, this would be "@ARM7TDMI@RVISS"



6. Save the configuration. It can now be used to run tests using the specified Debug Configuration for the simulator.

Unit Testing

C++test provides a unit testing Test Configuration designed specifically to work with RealView projects: Run RVDS 3.x and 4.x Tests. This Test Configuration is designed to run the previously-built test executable on a RealView Debugger and then collect its results. This Test Configuration is available in the **Builtin> Embedded Systems> ARM** category.

To run unit testing on target or simulator:

1. Manually create or automatically generate a set of test cases.
2. Duplicate the “Run RVDS 3.x 4.x Tests” test configuration as follows:
    a. Choose **Parasoft> Test Configurations**.
    b. Open the **Builtin> Embedded Systems> ARM** tree node.
    c. Right-click **Run RVDS 3.x 4.x Tests** and choose **Duplicate**.
3. Modify the Test Configuration as needed.
    - See General Configuration for details.
4. Select the desired testing context.
5. Run tests with the customized Test Configuration created above.

Debugging Test Cases

C++test does not support direct Test Cases debugging for this environment.

Use appropriate Debug/Launch Configuration for your original/tested project to load Test Executable and set breakpoints on wanted Test Cases manually

Application Monitoring

The "Builtin> Embedded Systems> ARM> Run RVDS 3.x 4.x Tests" Test Configuration is provided to facilitate Application Monitoring:

To run application monitoring on the target:

1. Duplicate the "Builtin> Embedded Systems> ARM> Run RVDS 3.x 4.x Tests" Test Configuration as follows:
    a. Choose **Parasoft> Test Configurations**.
    b. Open the **Builtin> Embedded Systems> ARM** tree node.
    c. Right-click **Run RVDS 3.x 4.x Tests** and choose **Duplicate**.
2. Modify the Test Configuration as needed.
    - See General Configuration for details.
3. Select the desired testing context.
4. Run tests with the customized Test Configuration created above.

Testing on the Simulator (RTSM Simulator) or Real Hardware

The test configurations provided with C++test are configured for testing with simulators or real hardware systems that allow remote execution via SSH protocol. These configurations assume tests are run on Linux-based applications. Testing non-Linux or bare metal applications is also possible. For more details please contact Parasoft support team.

Customizing Built-in Test Configurations

The Test Configurations provided to execute runtime tests for ARM DS-5 projects are accessible from the main menu: **Parasoft> Test Using> Builtin> Embedded Systems> ARM**. The built-in ARM configurations may require environment-specific customization. Typical customizations are done by configuring the test flow definition properties. To access the test flow definition properties, choose **Parasoft> Test Configurations> Your Test Configuration** and select the **Execution> General tab**. Make changes in the **Execution details** section.

The table below describes the test flow properties for the following ARM DS-5 built-in configurations:

- Run ARM Embedded Linux Test Executable
- Run ARM Embedded Linux Application with Memory Monitoring

Name	Default Value	Description
ARM Linux target	10.9.1.1	IP address of the target that will be used to execute tests for the tested project
ARM Linux target directory	/home/user	The directory on the target; where the test executable will be placed
ARM Linux target user name	user	The user name on the target

The table below describes the test flow properties for the following ARM DS-5 built-in configurations:

- Run DS-5 Test Executable (Software Model)
- Run DS-5 Application with Memory Monitoring (Software Model)

Name	Default Value	Description
Software Model Executable	FVP_EB_Cortex-A8	Simulator executable file

Users can easily alter the default test flow definition in cases where the default test's execution flow is insufficient (e.g. if additional steps are required to download the test executable to the target). For more details on customization of test flow definitions please refer to: Runtime Testing: Customizing the Test Execution Flow.

Unit Testing

The "Run ARM Embedded Linux Test Executable" and the "Run DS-5 Test Executable (Software Model)" test configuration facilitates the unit testing process. To run unit tesitng on the simulator or real hardware system:

1. Manually create or automatically generate a set of test cases.
2. Duplicate the unit test configuration.
3. Modify the test configuration as needed.
    - For details, see Customizing Built-in Test Configurations.
4. Select the desired testing context.
5. Run tests with your customized test configuration.

Debugging Test Cases

C++test does not support direct Test Cases debugging for this environment.

Use appropriate Debug/Launch Configuration for your original/tested project to load Test Executable and set breakpoints on wanted Test Cases manually.

Application Memory Monitoring

The "Run ARM Embedded Linux Application with Memory Monitoring" and "Run DS-5 Application with Memory Monitoring (Software Model)" test configurations facilitate the application memory monitoring process. To run application memory monitoring on the simulator or real hardware system:

- Duplicate the test configuration.
- Modify the test configuration as needed.
    - For details, see Customizing Built-in Test Configurations.
- Select the desired testing context.
- Run tests with the customized test configuration created above.