



User Guides

3.4 — Last update: Nov 26, 2020

Suresofttech

Table of Contents

1. Target Test Guides	2
1.1. Texas Instruments Code Composer Studio	3
2. Debugger User Guides	5
2.1. Lauterbach TRACE32	6
2.1.1. Supported target list that can generate cmm script automatically	7
2.1.2. Step1: Setting target environment in Controller Tester	8
2.1.3. Step2: Run the target test	9
2.1.4. Debug the target test	10
2.2. PLS Universal Debug Engine (UDE).....	11
2.2.1. Step1: Create a workspace in UDE IDE	12
2.2.2. Step2: Setting target environment in Controller Tester	14
2.2.3. Step3: Run the target test	15
2.2.4. Debug the target test	16
2.3. iSYSTEM winIDEA Debugger.....	17
2.3.1. Preparation for use of iSYSTEM winIDEA	18
2.3.2. Step1: Creating and setting up a winIDEA workspace	19
2.3.3. Step2: Setting target environment in Controller Tester	24
2.3.4. Step3: Run the target test	25
2.3.5. Debug the target test	26
2.4. IAR Embedded Workbench C-SPY Debugger	27
2.4.1. Step1: Creating an IAR embedded workbench project	28
2.4.2. Step2: Setting an IAR project.....	29
2.4.3. Step3: Setting target environment in Controller Tester	32
2.4.4. Step4: Run the target test	33
2.4.5. Debug the target test	34
2.5. Texas Instruments Code Composer Studio (CCS v4 and later).....	35
2.5.1. Step1: Create a project in Code Composer Studio	36
2.5.2. Step2 : Setting target environment in Controller Tester	38
2.5.3. Step3: Run the target test	41
2.5.4. Debug the target test	42
3. Target Build Guide	43
3.1. IAR Embedded Workbench IDE	44
3.2. Texas Instruments Code Composer Studio	46
3.3. CodeWarrior IDE.....	48
3.4. Hightec Development Platform IDE	49
3.5. Tasking VX IDE.....	50
3.6. Renesas CS+ IDE	51
3.7. MPLAB X IDE.....	53
3.8. Microsoft Visual Studio.....	54
3.9. GNU Compiler.....	55
4. Sharing Projects with Other Users	56

4.1. (Ver.3.3 or later) Guide to Share Projects.....	57
4.1.1. Export project	58
4.1.2. Import project.....	60
4.2. (Ver.3.2 or earlier) Guide to Share RTV Projects.....	65
4.2.1. Project sharing scenario.....	66
4.2.2. RTV server user guide	69
5. Identifying the Cause of a Test Error	70
6. (After Ver.3.4) Source Code Modification and Test Reconfiguration.....	72
6.1. Run [Test Reconfiguration].....	73
6.2. In Cases of Detected Modification Automatically	77
6.3. In Cases of Undetected Modification Automatically	80

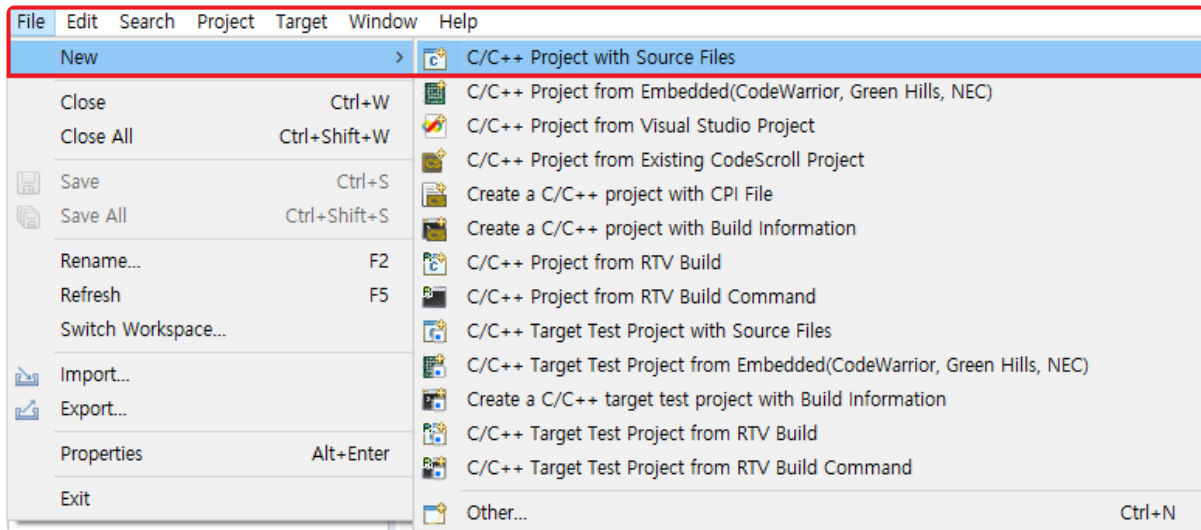
1. Target Test Guides

This user guides document describes how to execute target tests using CodeScroll Controller Tester.

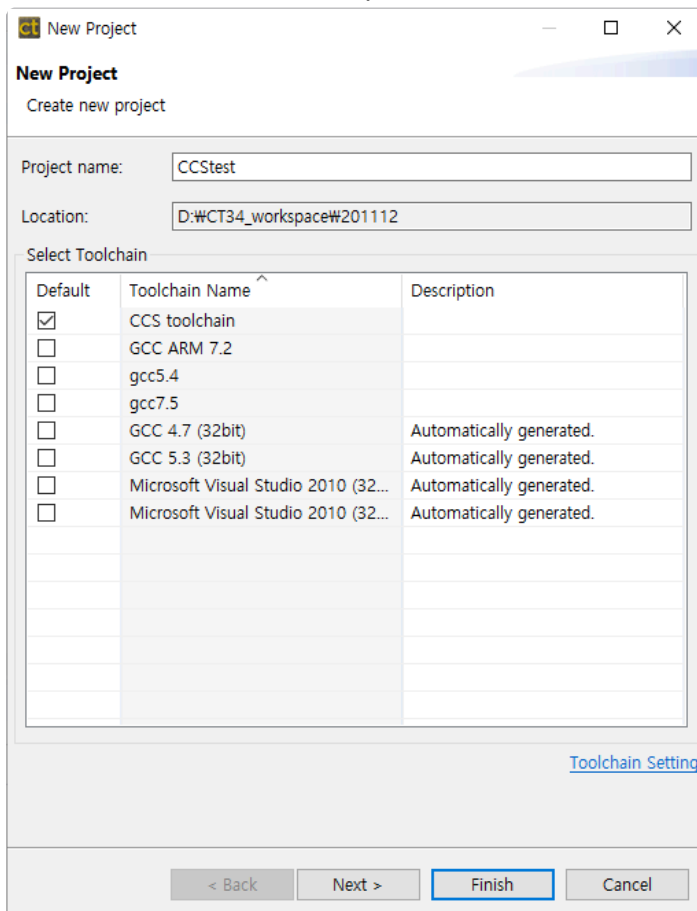
[Texas Instruments Code Composer Studio](#)

1.1. Texas Instruments Code Composer Studio

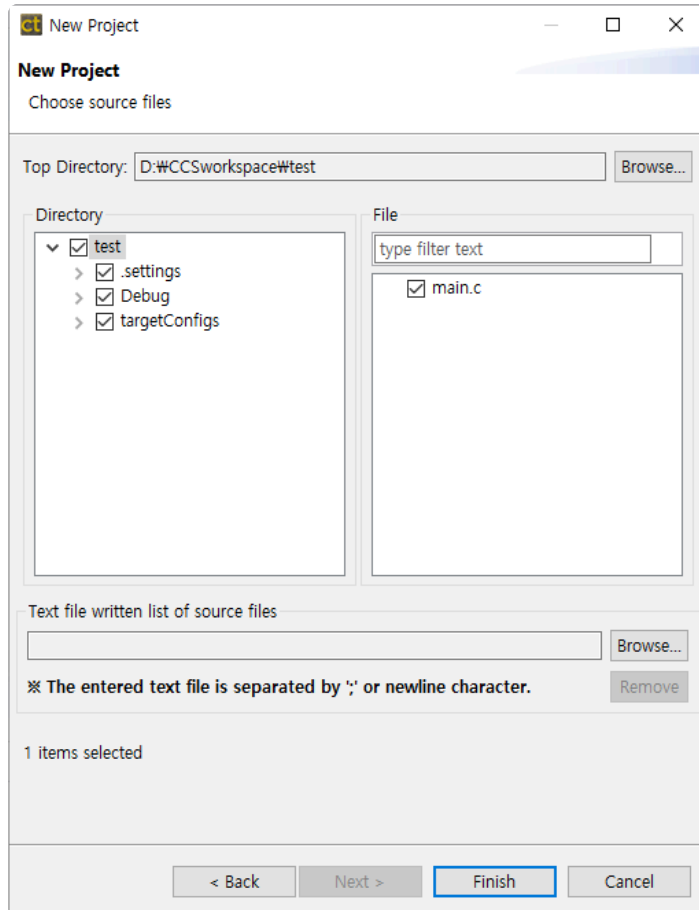
1. Create a CodeScroll Controller Tester project.



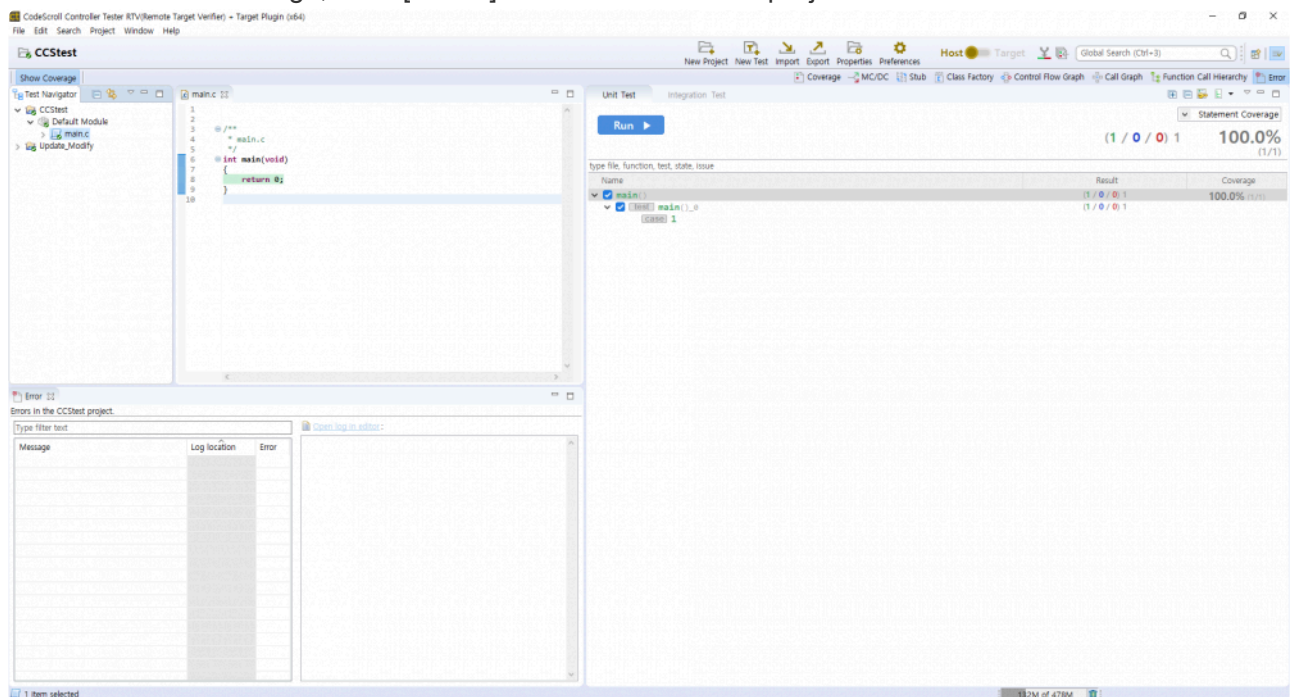
2. Select a created Code Composer Studio toolchain.



3. Select source files to test.



4. When finish the settings, click [Finish] button to create the project.



5. To use debuggers, set up in Code Composer Studio and CodeScroll Controller Tester. For more information, refer to [Texas Instruments Code Composer Studio](#), a sub-topic of [Controller Tester Debugger User Guides](#) in this document.

2. Debugger User Guides

This user guides document describes how to use debugger when executing CodeScroll Controller Tester target test.

- [Lauterbach TRACE32](#)
- [PLS Universal Debug Engine](#)
- [iSYSTEM winIDEA Debugger](#)
- [IAR Embedded Workbench C-SPY Debugger](#)
- [Texas Instruments Code Composer Studio](#)

2.1. Lauterbach TRACE32

Controller Tester can target test using the TRACE32 debugger.

Controller Tester uses TRACE32's cmm script to run tests in the target environment and get the results.

A list of targets supported by TRACE32 can be found on the [Lauterbach homepage](#).

- [Supported target list that can generate cmm script automatically](#)
- [Step1: Setting target environment in Controller Tester](#)
- [Step2: Run the target test](#)

2.1.1. Supported target list that can generate cmm script automatically

Controller Tester automatically generates a cmm script file or receives it from the user.

If the cmm script can be generated automatically, you only need to enter the chip name of the target. If you cannot generate cmm scripts automatically, you must enter the cmm script file path manually.

The targets that currently support the automatic generation of cmm scripts are:

PowerPC	mpc5554, mpc5553, mpc5534, mpc556x, mpc551x, mpc560xe, spc560bxx, spc560pxx, spc560sxx, mpc560xb, mpc560xp, mpc560xs, spc563m54, mpc5632m, spc563m60, mpc5633m, spc563m64, mpc5634m, mpc564xs, mpc5668, mpc5674, mpc5644a, spc564a80, mpc5642a, spc564a70, mpc567xk, spc56hk, mpc5643l, spc56el60, spc56el70, mpc5644b, mpc5644c, spc564b64, spc56ec64, mpc5645b, spc564b70, mpc5645c, spc56ec70, mpc5646b, spc564b74, mpc5646c, spc56ec74, mpc5676r, spc56ap, mpc5746m, mpc5744k, spc574k74, mpc5777m, spc57hm90, mpc574xp, mpc574xg, mpc574xr, mpc577xk, mpc5777c, spc570s, mpc5726l, spc572l, spc574s, spc58ne, spc58eg, spc58nn, spc582b, spc58ec, spc58nh, spc584b, s32r274, s32r264, s32r372
ARM	mkw01, mkw20, mkv30, mkv40, mkv10, mkv50, mkm30, mkl0, mkl10, mkl20, mkl30, mkl40, mkl80, mk0, mk10, mk20, mk30, mk40, mk50, mk60, mk70, mk80, mac57d54h, mac71×1, mac71×2, mac71×4, mac71×5, mac71×6, mac72×1, lpc51u68, lpc54xx, lpc8xx, lpc11xx, lpc12xx, lpc13xx, lpc17xx, lpc18xx, lpc21xx, lpc22xx, lpc23xx, lpc24xx, lpc28xx, lpc29xx, lpc40xx, lpc43xx, imxrt1064, xmc1100, xmc1200, xmc1300, xmc1400, xmc4100, xmc4200, xmc4300, xmc4400, xmc4500, xmc4700, xmc4800, tle98, s3fm02g, s32k, s6e1a, s6e1c, s6j3
tricore	tc2dx, tc21x, tc22x, tc23x, tc26x, tc27x, tc29x, tc35x, tc37x, tc38x, tc39x, tc116x, tx1167, tx1197, tc1724, tc1728, tc1736, tc1762, tc1764, tc1766, tc1767, tc1782, tc1784, tc1791, tc1792, tc1793, tc1796, tc1797, tc1798

2.1.2. Step1: Setting target environment in Controller Tester

Select Debugger on the target environment setting page of the Controller Tester. Only a list of debuggers supported is displayed, depending on the toolchain selected for the project.

Set the debugger to TRACE32.

► Freescale ► CodeWarrior-MPC55xx ► 2.6 ► others ► trace32

The setting items are displayed according to the selected information. The items you need to set when using the TRACE32 debugger are shown in the table below.

Some of the settings are required.

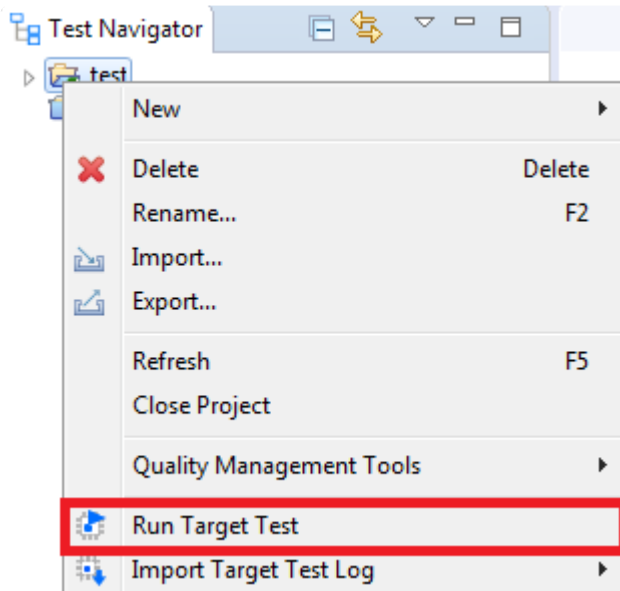
trace32_exe_file_path	TRACE32 executable file path. Each target has a different executable file, so you need to make sure that the target executable is the correct one. Required
target_binary_path	Path to the binary file for loading into the target environment. Check and enter the path where the target binary file is created in your IDE or build script. Required.
chip	Enter the chip name of the target you are using. It is used when auto-generating a cmm script, so you need to enter the correct chip name.
user_defined_cmm_script_file_path	Custom cmm script file path. For targets that do not support automatic generation of cmm scripts, you must write a script to set the debugger and target usage environment, or enter the path to the cmm script file you are using.

2.1.3. Step2: Run the target test

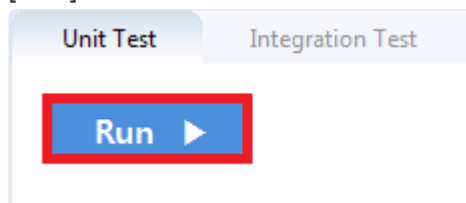
You must exit the running TRACE32 program before running the target test.

You can run a target test by selecting [Run Target Test] from the project context menu in the Test Navigator view or by clicking the [Run] button in the Test View.

- [Run Target Test]



- [Run]



* When you run the target test, the TRACE32 program runs. If the test is succeeded, the TRACE32 program ends automatically.

2.1.4. Debug the target test

1. After setting it as a target, right-click the test case in the 'Unit Test' view and click 'Check Debug Information'
2. Build the user project directly or execute the build script registered in the 'target environment' setting in the controller tester project
3. Verify that the build was successful
4. Restore the original source by opening the project in Controller Tester
5. After running Trace32, open the cmm script file (start.cmm) and execute 'debug' (Controller_Tester_project_path/.csdata/target/start.cmm)
6. Click the 'step' button to go to the first line of the target.cmm script
7. Add breakpoint to 'Go.Hll' in target.cmm file
8. Click 'Var' > 'Show Function'
9. Double-click after searching for the function to be tested
10. Add breakpoint at the beginning of the function
11. Click the 'step' button and confirm that the debugging point moves to the location specified in step 10.
12. 'Var' > 'Show Local...' . Click to confirm that the value of the local variable changes
13. Run up to the debugging point

2.2. PLS Universal Debug Engine (UDE)

Controller Tester can target test using the UDE debugger.

Controller Tester uses debugging scripts supported by UDE to run tests and get results in the target environment.

A list of targets available for connection to UDE can be found on the [PLS homepage](#).

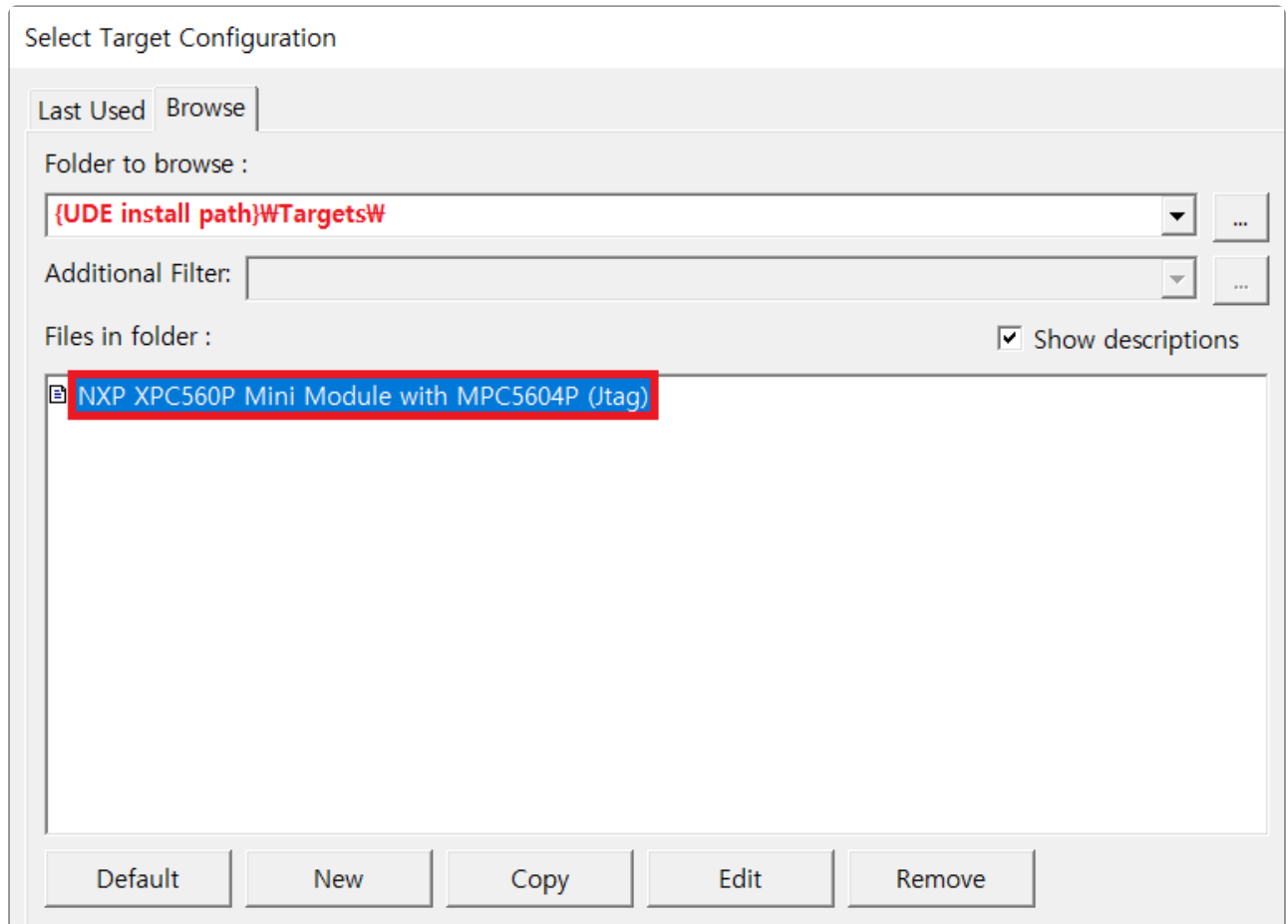
Controller Tester uses the UDE workspace information to perform target tests. For this reason, users must first create a workspace before performing a target test.

- [Step1: Create a workspace in UDE IDE](#)
- [Step2: Setting target environment in Controller Tester](#)
- [Step3: Run the target test](#)

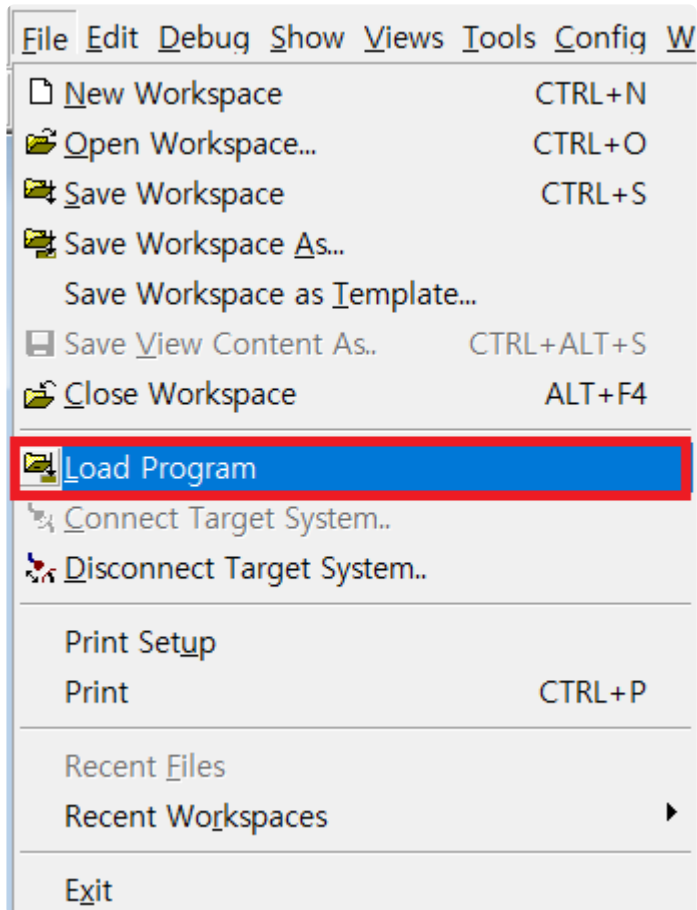
2.2.1. Step1: Create a workspace in UDE IDE

UDE can generate UDE workspaces from the UDE desktop IDE.

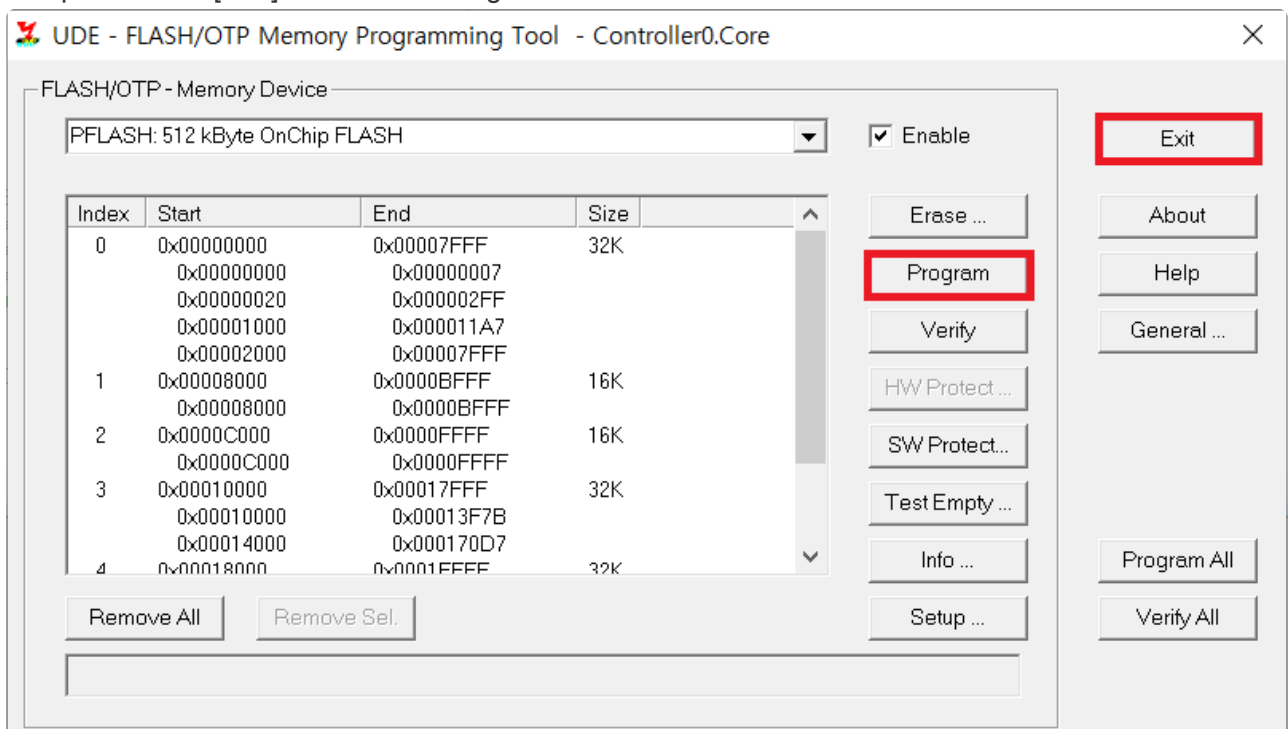
1. Create the workspace by selecting the configuration file suitable for the target used.



2. Click the [File]> [Load Program] button to load the binary file. At this point, select the binary file built from the test code.



- Follow the instructions and press the [program] button to load the binary file into the target according to the target settings. If the load completes successfully, the workspace setup is complete. Click [Exit] to exit the dialog.



See the manual provided by UDE for details.

2.2.2. Step2: Setting target environment in Controller Tester

Select Debugger on the Target Environment configuration page of the Controller Tester. Only a list of debuggers supported is displayed, depending on the toolchain selected for the project.

Set the debugger to UDE.

► Freescale ► CodeWarrior-MPC55xx ► 2.6 ► others ► ude

The setting items are displayed according to the selected information. The items you need to set when using the UDE debugger are shown in the table below.

Some of the settings are required.

target_binary_path	Path to the binary file for loading into the target environment. Check and enter the path where the target binary file is created in your IDE or build script. Required.
ude_project_file	Path to the workspace project file (.wsx) generated by the UDE IDE. Required.

The default scripting language used by Controller Tester is visual basic script.

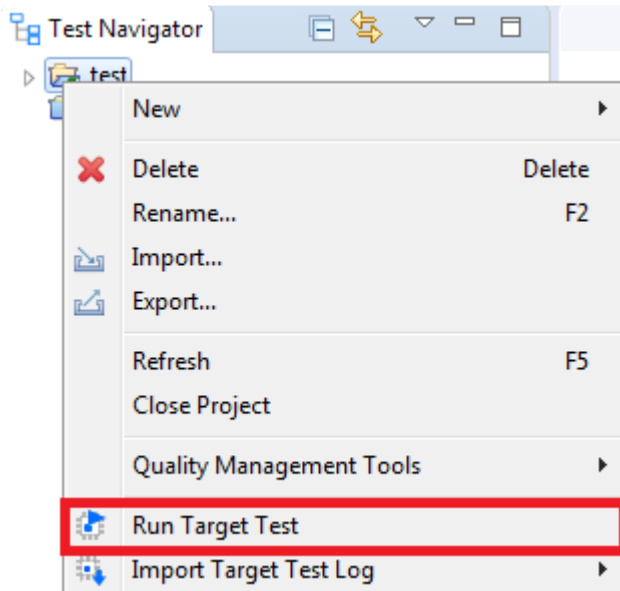
When the target configuration is complete, click the [OK] or [Finish] button. You are ready to execute the target test.

2.2.3. Step3: Run the target test

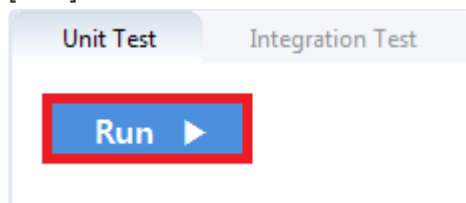
You must have exited the UDE desktop IDE to run the target test.

You can run a target test by selecting [Run Target Test] from the project context menu in the Test Navigator view or by clicking the [Run] button in the Test View.

- [Run Target Test]



- [Run]



* UDE debugging scripts can be written in languages such as C ++, .NET, and Perl. See the UDE Automation Basics documentation included in the UDE manuals for other supported languages that can be scripted.

2.2.4. Debug the target test

1. After setting it as a target, right-click the test case in the 'Unit Test' view and click 'Check Debug Information'
2. Build the user project directly or execute the build script registered in the 'target environment' setting in the controller tester project
3. Verify that the build was successful
4. Restore the original source by opening the project in Controller Tester
5. Select project after executing PIs Ude (.wsx file)
6. Select the output file built in step.2
7. Notice that the source file and function information contained in the output file are displayed on the left navigation.
8. Select a source file containing the function to be tested and add breakpoints in the function
9. Press F5 to start from the entry point

2.3. iSYSTEM winIDEA Debugger

Controller Tester provides the ability to run tests on your target environment and get results from it automatically by using winIDEA debugging scripts.

The list of targets supported by winIDEA can be found on the [iSYSTEM home page](#).

The execution of the debugging script requires the python SDK installed together when installing winIDEA. If it is not installed, you can download it from the [iSYSTEM SDK installation page](#). Also, you should check the version of winIDEA you use if it supports the SDK. The debugging script provided by Controller Tester is based on python 3.3.

This document describes the process from creating a project in winIDEA to running a target test in Controller Tester. The iSYSTEM BlueBox iC5000 Unit debugger and NXP's MPC56xx target are used for the examples.

- [Preparation for use of iSYSTEM winIDEA](#)
- [Step1: Creating and setting up a winIDEA workspace](#)
- [Step2: Setting target environment in Controller Tester](#)
- [Step3: Run the target test](#)


2.3.1. Preparation for use of iSYSTEM winIDEA

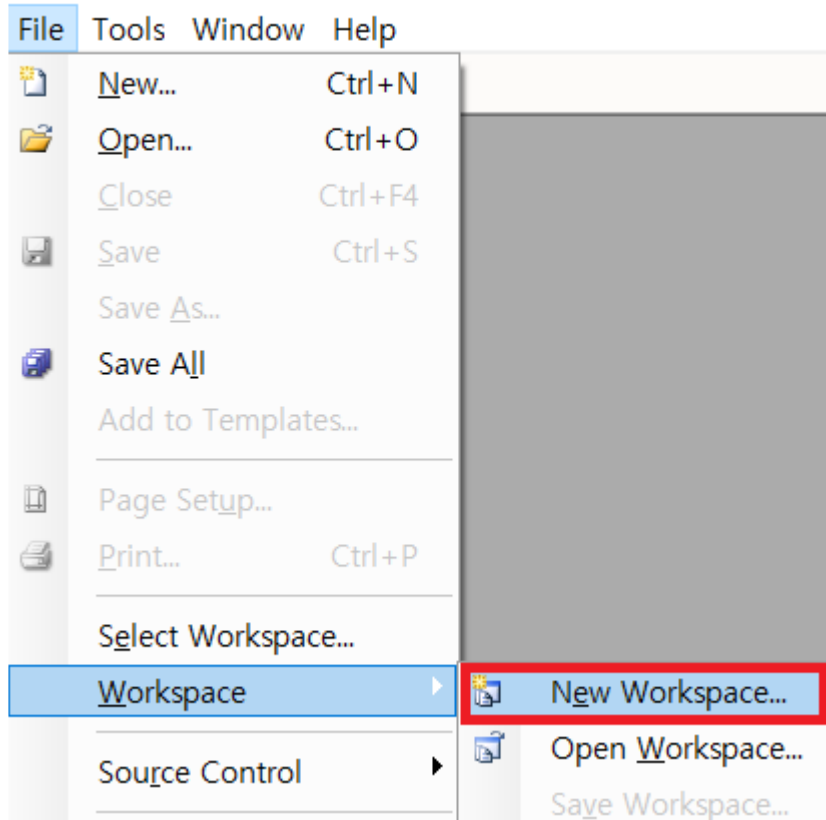
Target testing with winIDEA in Controller Tester requires a debugger that winIDEA supports.

Before running the target test, you need to create a winIDEA workspace and connect the debugger for use to the PC with Controller Tester.

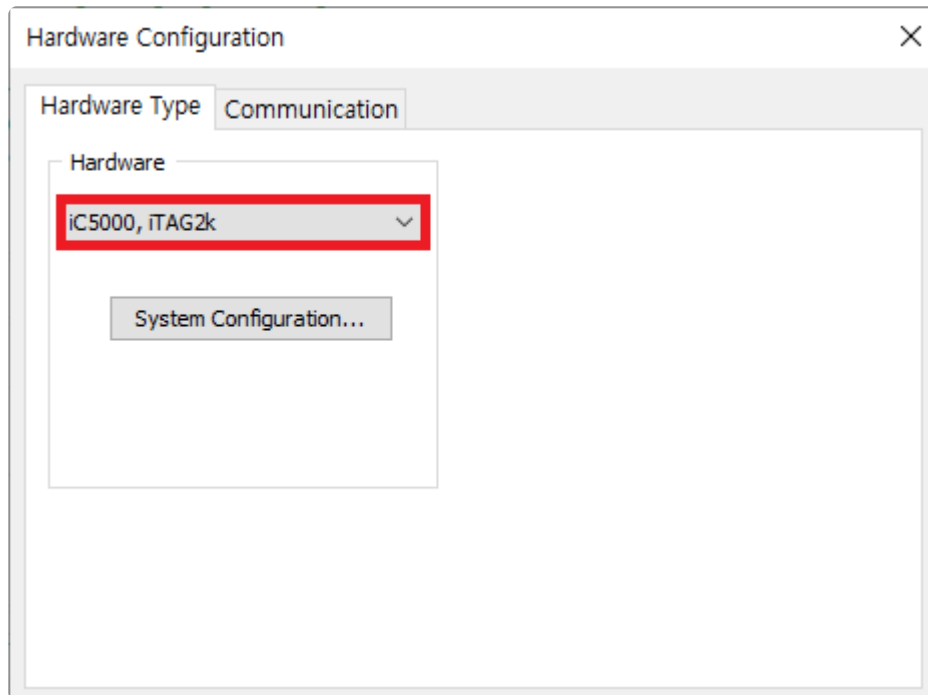
2.3.2. Step1: Creating and setting up a winIDEA workspace

1. After running winIDEA, create a new workspace by selecting [File]> [Workspace]> [New Workspace ...] from the top menu. Additional workspace settings are required to use the workspace you create for the Controller Tester target test.

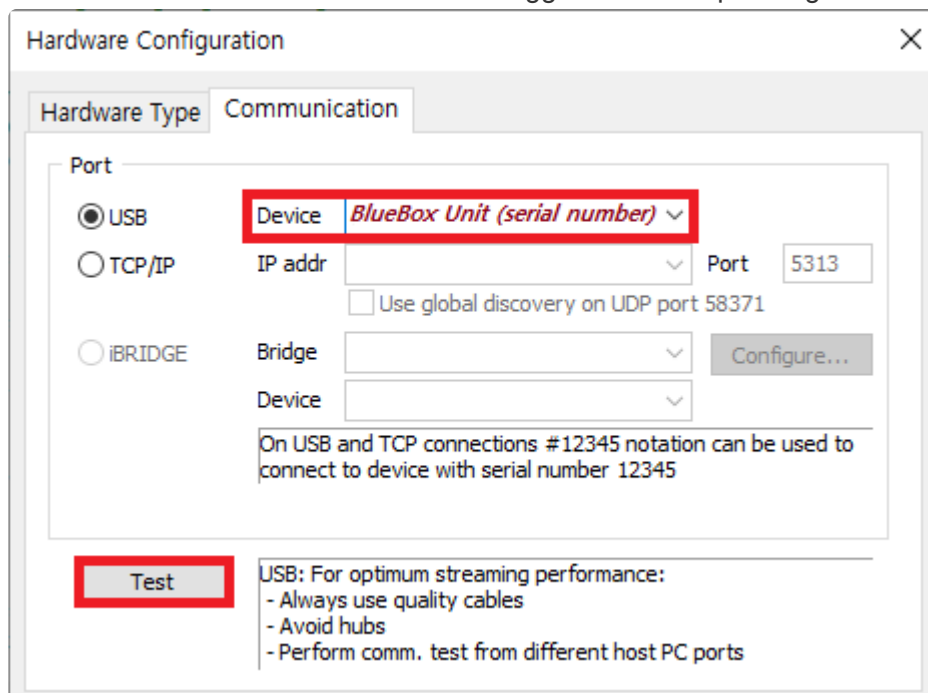
 winIDEA



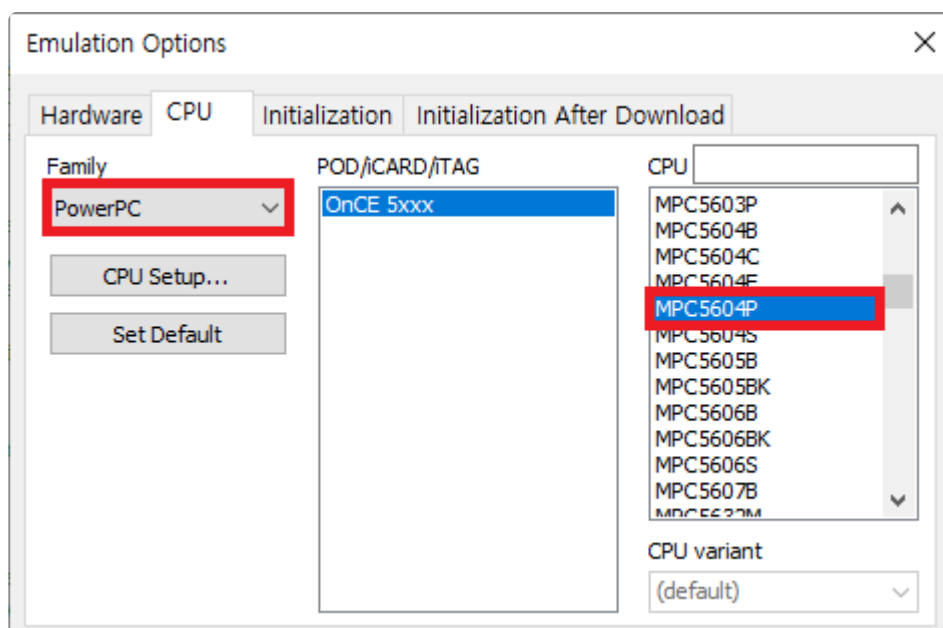
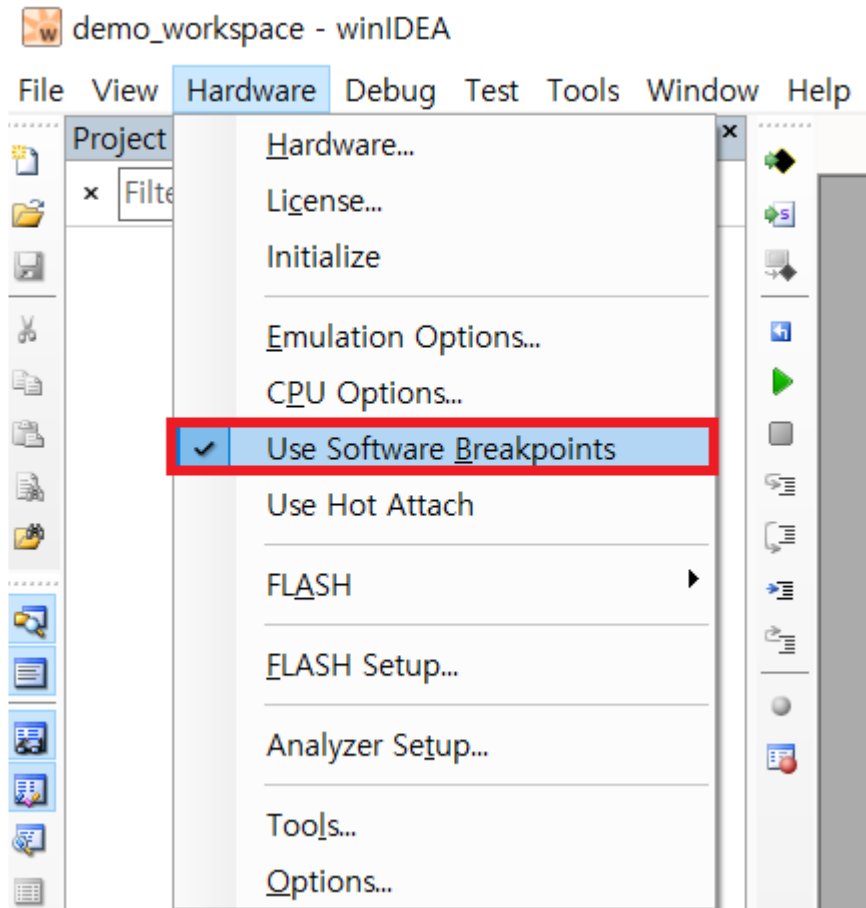
2. First, go to the top menu, select [Hardware]> [Hardware...], and then select the type of the connected BlueBox in the [Hardware Type] tab.



3. Next, set the communication method in the [Communication] tab, and press the [Test] button to check the connection to the debugger. Please refer to the iSYSTEM BlueBox manual for instructions on how to connect the debugger device depending on the communication method.

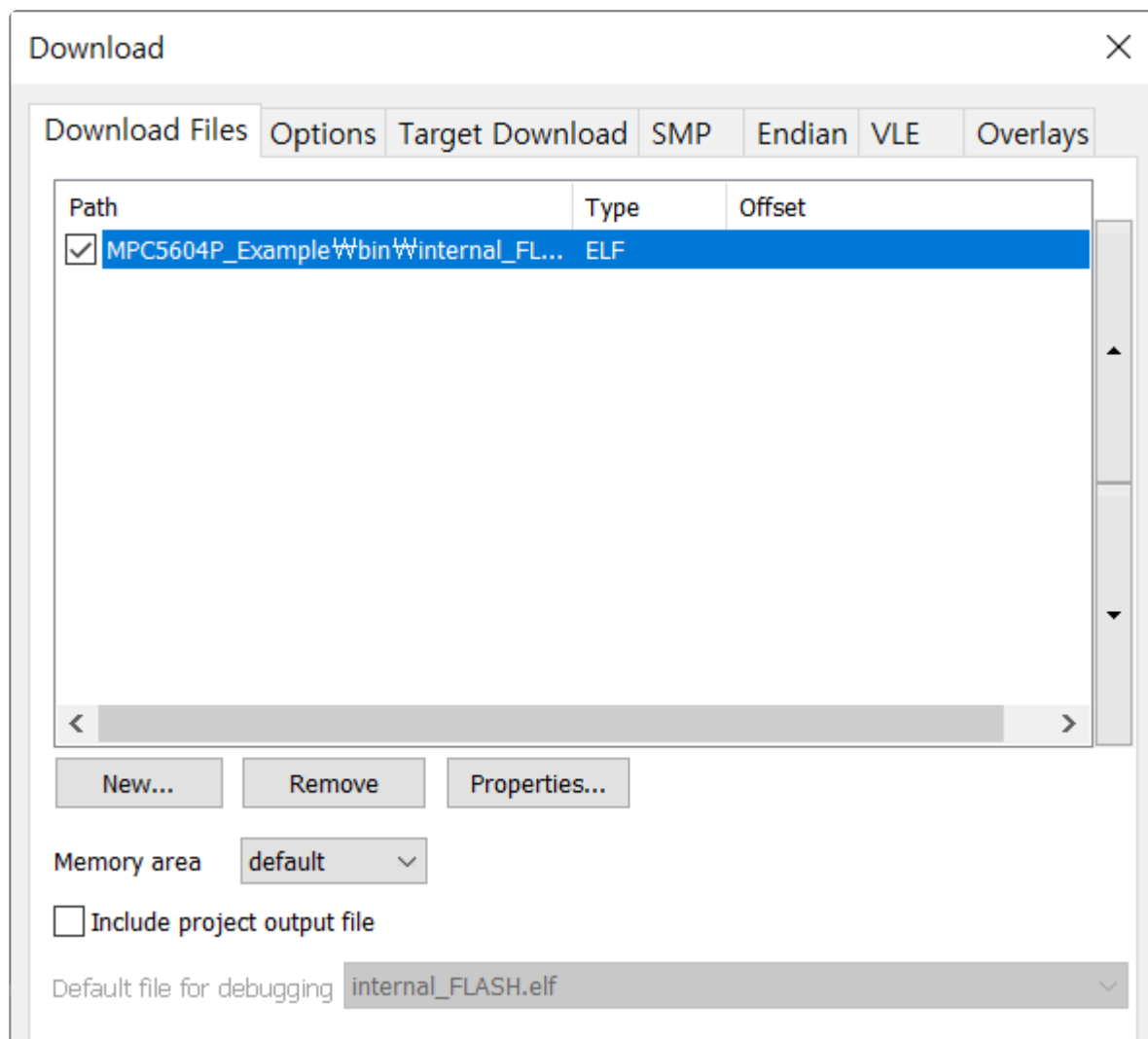


4. Click [Hardware]> [Use Software Breakpoints] on the top menu to activate it, and then select the target type to use in the [CPU] of [Hardware]> [Emulation Options...].

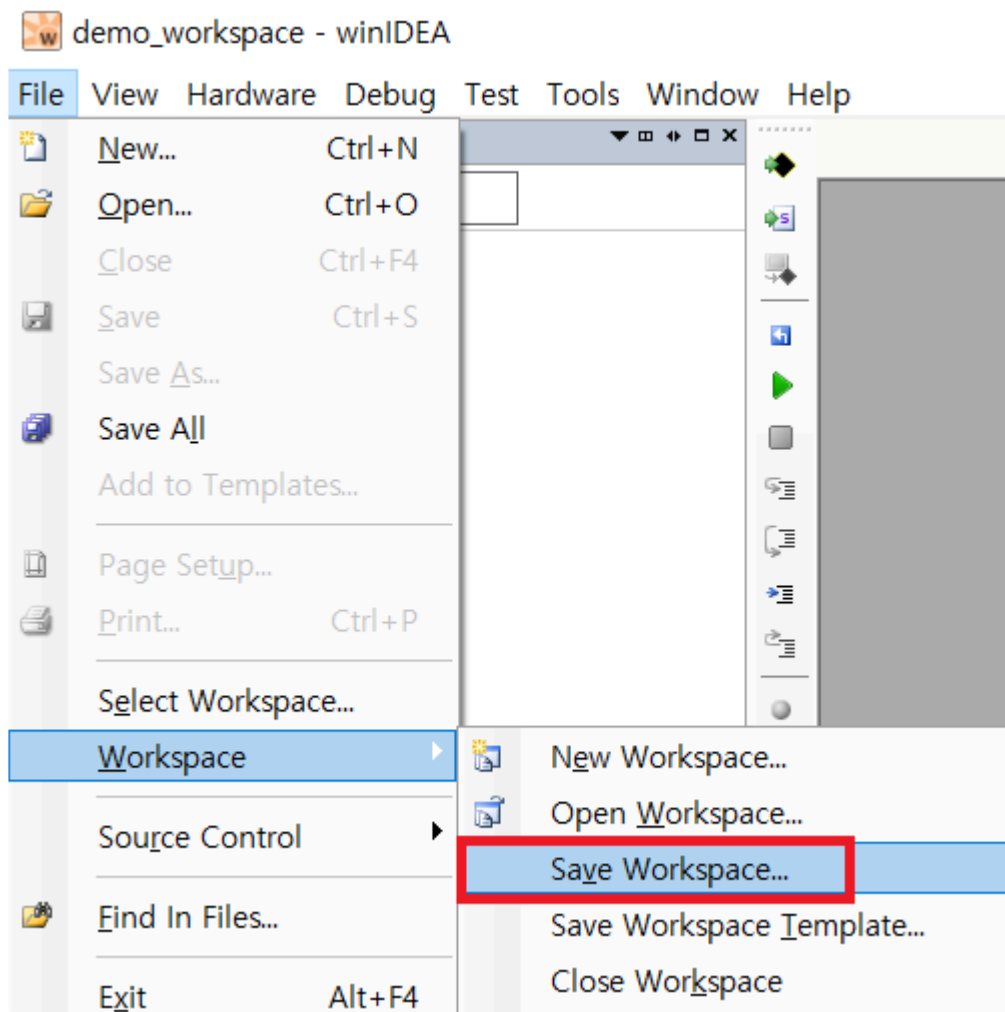


* The specific options you need to set for each target may vary.

- After the debugger setup is complete, you need to register the binary path of the software under test in the workspace. First, build the source code under test to generate the binary. Then from winIDEA's top menu [Debug]> [Files for Download ...], select [New...] and add the binary generated.



- When everything is set up, save the workspace to create a winIDEA workspace file (.xjrf). The workspace file is used to configure the target test using winIDEA in Controller Tester.



You are now finished creating the winIDEA workspace for the target test.

2.3.3. Step2: Setting target environment in Controller Tester

Select a debugger in the [New Project] wizard of the target test project or [Target environment settings] of the project properties on Controller Tester. The list of selectable debuggers depends on the toolchain selected for the project.

Set the debugger to BlueBox.

► Freescale ► CodeWarrior-MPC55xx ► 2.6 ► others ► bluebox

The fields to be set are displayed according to the selection. If you are using BlueBox, the fields are shown in the table below.

Required fields are displayed in red in Controller Tester.

winidea_binary_path	The winIDEA execution file(winIDEA.exe) path. Required.
winidea_workspace_file_path	The path of the workspace file (.xjrf) created by winIDEA. Required.

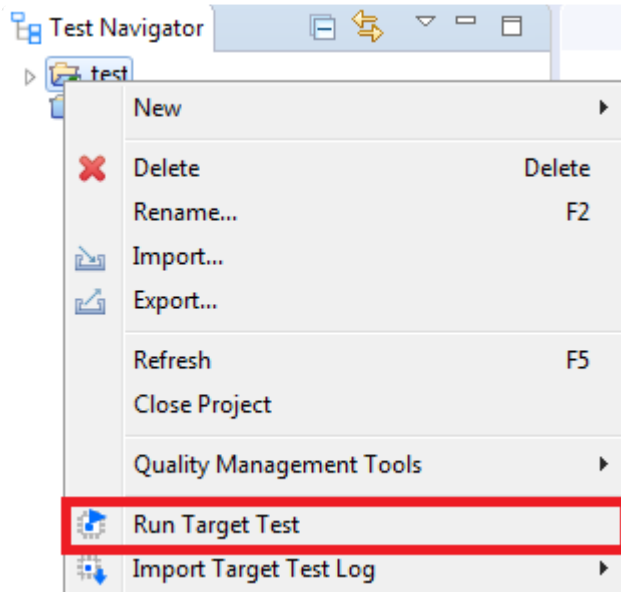
The default scripting language provided by Controller Tester is python. If you use a custom debugging script, you need to write it in python to work properly. If you write in other languages, refer to the [iSYSTEM homepage](#) to install additional SDKs.

When the target environment settings are complete, click the [OK] or [Finish] button. Now you are ready to run the target test.

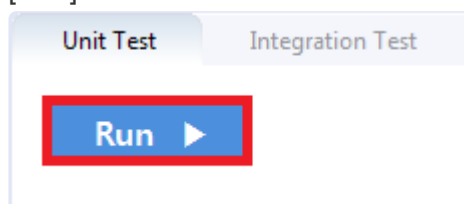
2.3.4. Step3: Run the target test

You can run a target test by selecting [Run Target Test] from the project context menu in the Test Navigator view or by clicking the [Run] button in the Test View.

- [Run Target Test]



- [Run]



* Target tests cannot be run if winIDEA is running. You must exit winIDEA before running the target test in Controller Tester.

2.3.5. Debug the target test

1. After setting it as a target, right-click the test case in the 'Unit Test' view and click 'Check Debug Information'
2. Build the user project directly or execute the build script registered in the 'target environment' setting in the controller tester project
3. Verify that the build was successful
4. Restore the original source by opening the project in Controller Tester
5. After running winIDEA, select the workspace containing the built project (.xjrf file)
6. Download to binary file target by selecting [Debug]> [Download]
7. Debugging mode by pressing the Run button at the top
8. Double-click [Project]> [Functions], move to the function location, and set the debugging point where you want
9. Press F5 to proceed debugging

2.4. IAR Embedded Workbench C-SPY Debugger

Controller Tester provides the ability to automatically run tests and get results in the target environment through the IAR Embedded Workbench C-SPY debugging function.

The list of targets supported by C-SPY can be found on the [IAR website](#).

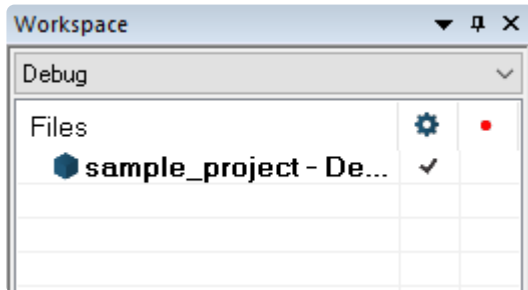
To test a target with the IAR Embedded Workbench C-SPY in the Controller Tester, you need a C-SPY compatible debugging probe. You need to create an IAR Embedded Workbench project and connect the debugging probe to be used with the PC where Controller Tester is installed before performing the target test.

The list of debugging probes provided by IAR can be found on the [homepage](#).

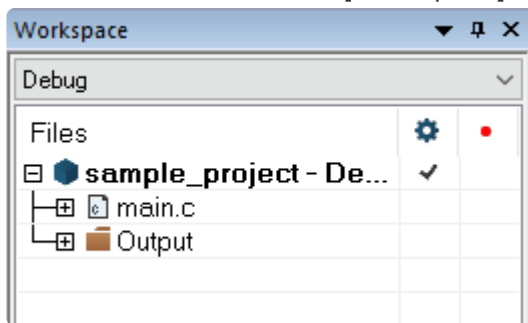
- [Step1: Creating an IAR embedded workbench project](#)
- [Step2: Setting an IAR project](#)
- [Step3: Setting target environment in Controller Tester](#)
- [Step4: Run the target test](#)

2.4.1. Step1: Creating an IAR embedded workbench project

1. Click [File]> [New Workspace] to create a new workspace and then click [Project]> [Create New Project...] to create a project file (.ewp). When a project file created, the project name is displayed in the [Workspace] view of the IAR Embedded Workbench.



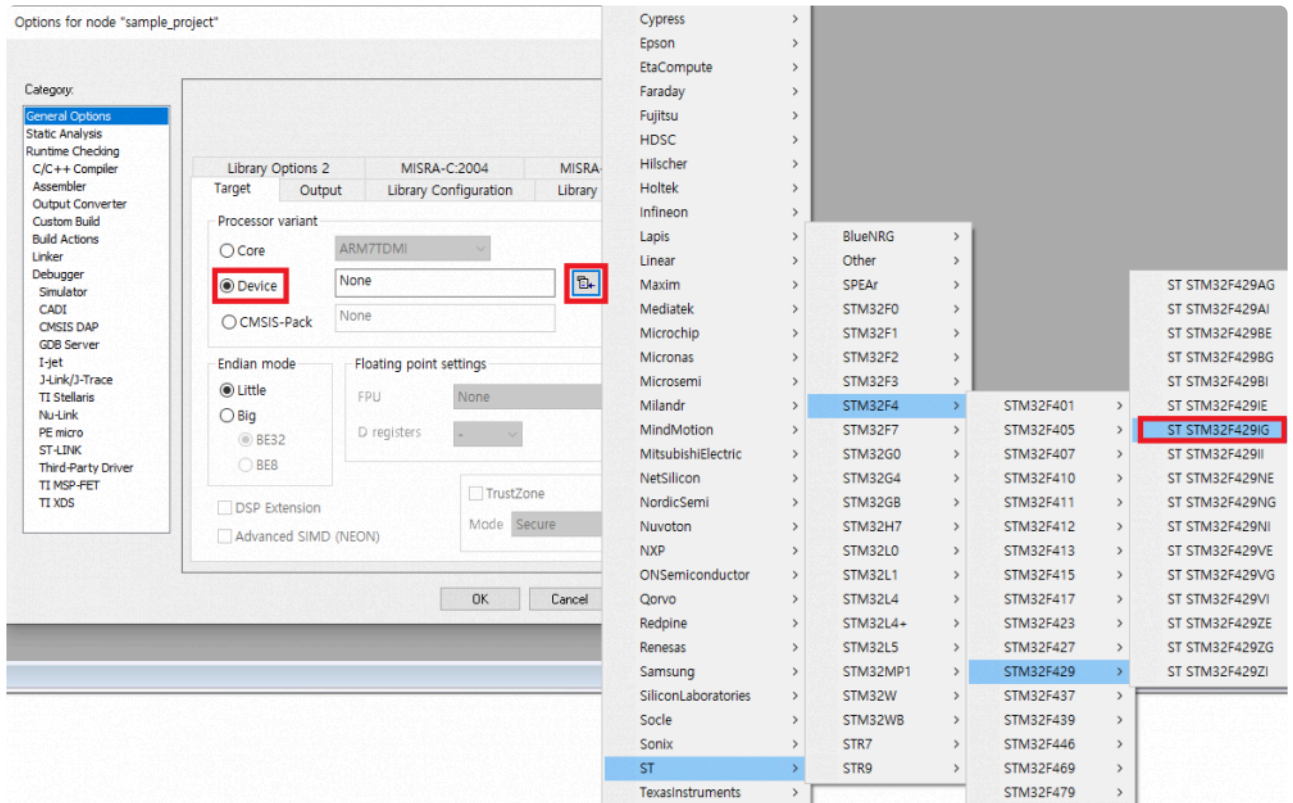
2. Next, you need to add the source files under test to the project. Right-click on the project, click [Add]> [Add Files...] and add the source files to be tested. The added source files are displayed in a hierarchical structure in [Workspace] view.



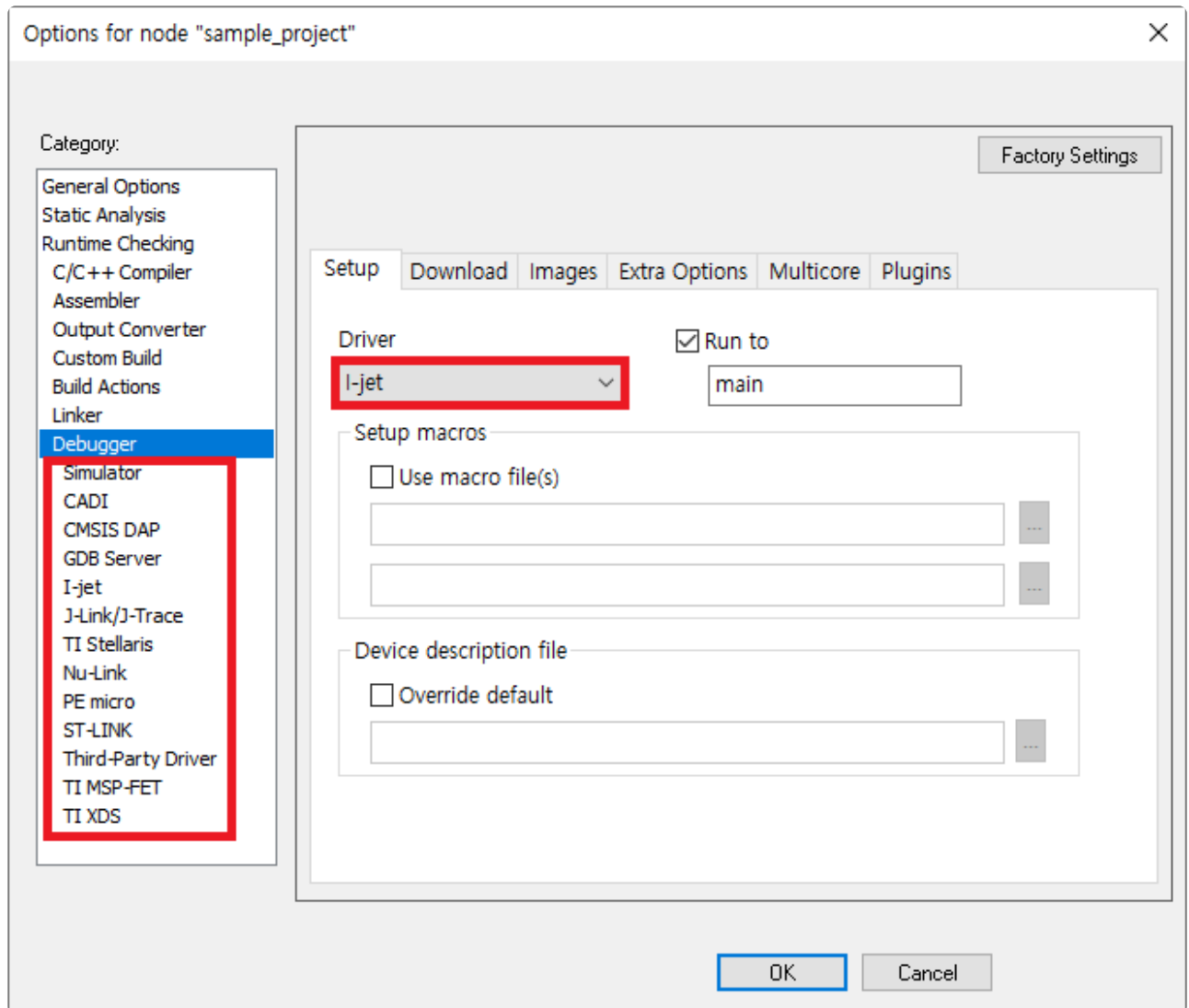
2.4.2. Step2: Setting an IAR project

If you created a project, you need to configure the project to use the C-SPY debugging feature. Right-click on the created project and select [Options ...].

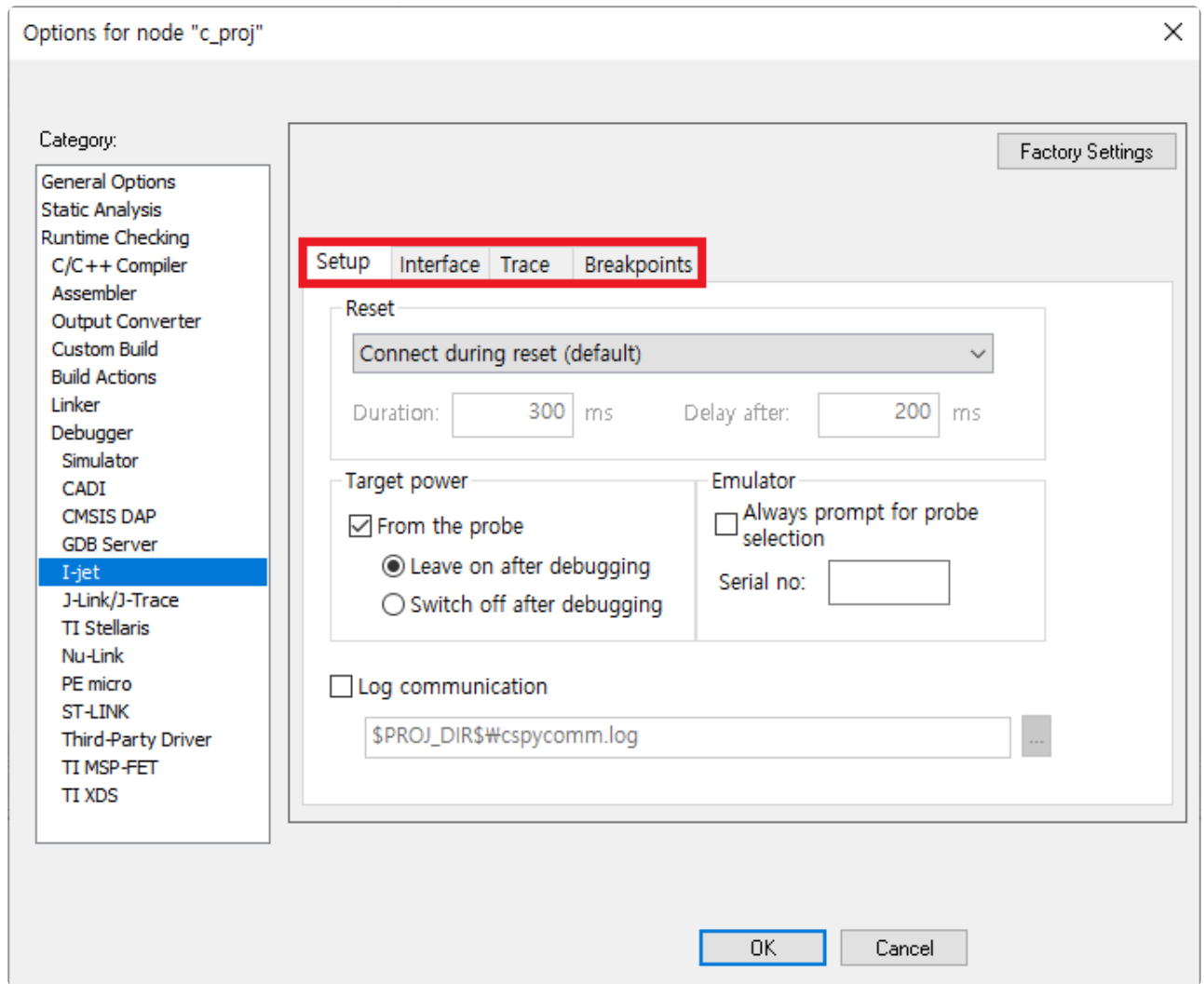
1. First, set [Processor variant] in [General Options]. For example, for ARM's STM32F429IG target, select Device and select a name that matches the target from the target list on the right.



2. Second, go to the category [Debugger] and select the debugging probe you want to use in the [Driver] field. Set the details in the Debugging Probe section at the bottom of the [Debugger] category, depending on how the selected debugging probe and PC are connected.



3. If I-jet is selected, select [I-jet] at the bottom of the [Debugger] category to set details. For a description of each setting tab, refer to the IAR debugger manual you want to use.



Now you are done creating and setting the IAR project for target testing.

2.4.3. Step3: Setting target environment in Controller Tester

Select a debugger in the [New Project] wizard of the target test project or [Target environment settings] of the project properties on Controller Tester. The list of selectable debuggers depends on the toolchain selected for the project.

When creating a project using the IAR toolchain, the debugger must be set to ide to use the IAR C-SPY debugging feature.

► IAR ► ARM-Compiler ► 5.x ► others ► ide

The fields to be set are displayed according to the selection. The fields for C-SPY are as shown in the table below.

Required fields are displayed in red in Controller Tester.

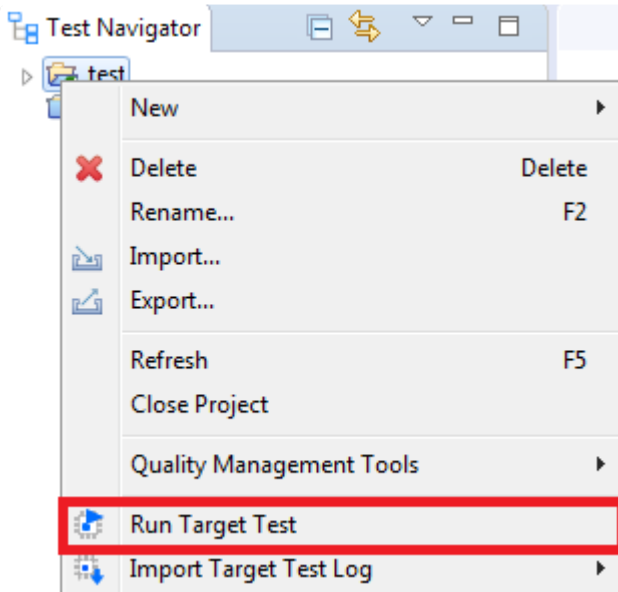
cspy_debug_general_xcl_file_path	The path to the debug.general.xcl file required when using the IAR Embedded Workbench C-SPY debugger. When creating an IAR project, the project file (.ewp) is automatically created in the [setting] folder in the saved location. Required.
cspy_debug_driver_xcl_file_path	Path to the debug.driver.xcl file required when using the IAR Embedded Workbench C-SPY debugger. When creating an IAR project, the project file (.ewp) is automatically created in the [setting] folder in the saved location. Required.

When the target environment settings are complete, click the [OK] or [Finish] button. Now you are ready to run the target test.

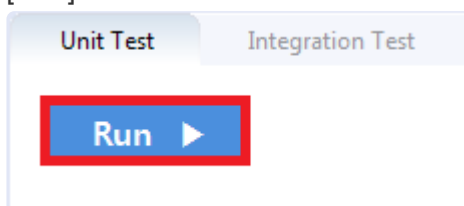
2.4.4. Step4: Run the target test

You can run a target test by selecting [Run Target Test] from the project context menu in the Test Navigator view or by clicking the [Run] button in the Test View.

- [Run Target Test]



- [Run]



2.4.5. Debug the target test

1. After setting it as a target, right-click the test case in the 'Unit Test' view and click 'Check Debug Information'
2. Build the user project directly or execute the build script registered in the 'target environment' setting in the controller tester project
3. Verify that the build was successful
4. After IAR Workbench run, select the workspace containing the built project (.eww file)
5. Select the source file with the function to be tested in the workspace view, and click the left side of the line to add the debugging point
6. Right-click the project in the workspace view and open 'Options ...' to check the Run to option check in the Debugger item and check that it is designated as 'main'
7. Click the Download and Debug button at the top to start from main
8. Press F5 to proceed to the debugging point to debug

2.5. Texas Instruments Code Composer Studio (CCS v4 and later)

Controller Tester can run target tests using the Code Composer Studio debugger. Controller Tester uses debugging scripts supported by Code Composer Studio (since version 4.x) to run the tests in target environment and get results. Check the Code Composer Studio manual for a list of debugging devices you can connect to and use with Code Composer Studio.

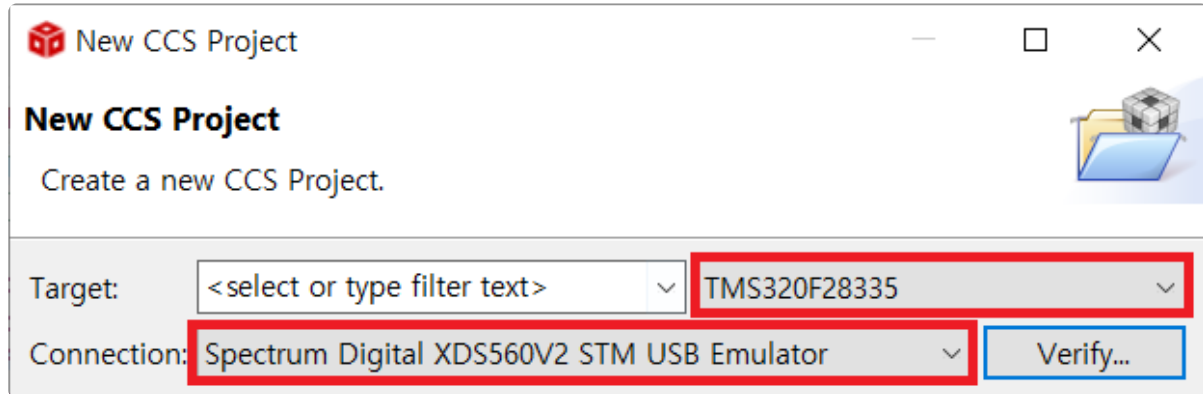
This document describes how to use Code Composer Studio debugger with following three steps.

- [Step1: Create a project in Code Composer Studio](#)
- [Step2 : Setting target environment in Controller Tester](#)
- [Step3: Run the target test](#)

The example uses Spectrum Digital's XDS560v2 as a debugger and Texas Instruments' TMS320 as target device.

2.5.1. Step1: Create a project in Code Composer Studio

1. Run Code Composer Studio and create a new project. Select [File]-[New] from the top menu and select the desired project type. In this case, click [CCS Project] to create a project. After entering the target and debugger information used, click [Verify] to confirm that the connection is successful.



2. After verifying the debugger and target connections, enter the remaining settings. The example uses the C2000 Ti compiler. When you click [Finish], the CCS project is created in the workspace.

C28XX [C2000]

Project name:

☒ Use default location

Location:

Compiler version:

▶ Tool-chain

▼ Project templates and examples

type filter text

- ▼ Empty Projects
 - Empty Project
 - Empty Project (with main.c)
 - Empty Assembly-only Project
 - Empty PowerSuite Project
 - Empty RTSC Project

Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Open [Resource Explorer](#) to browse a wide selection of example projects...

Open [Import Wizard](#) to find local example projects for selected device...

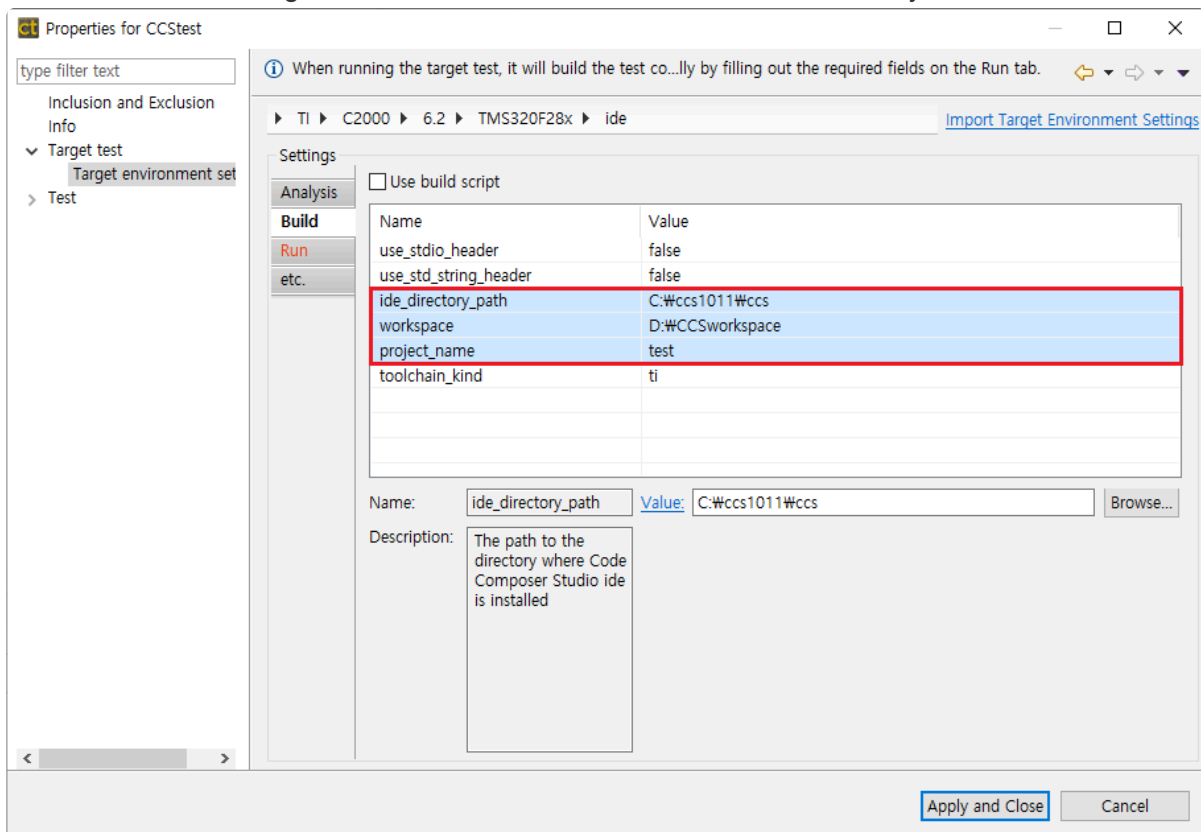
Code Composer Studio supports several more debuggers in addition to the built-in debuggers from Texas Instruments.

1. TI XDS USB (Code Composer Studio default)
2. BlackHawk JTAG emulator
3. Spectrum digital
4. MSP430 USB
5. MSP432 USB
6. Tiva/Stellaris ICDI

Controller Tester controls the debugger supported by Code Composer Studio with javascript. You can select the target and debugger details from the Project Settings screen in Code Composer Studio.

2.5.2. Step2 : Setting target environment in Controller Tester

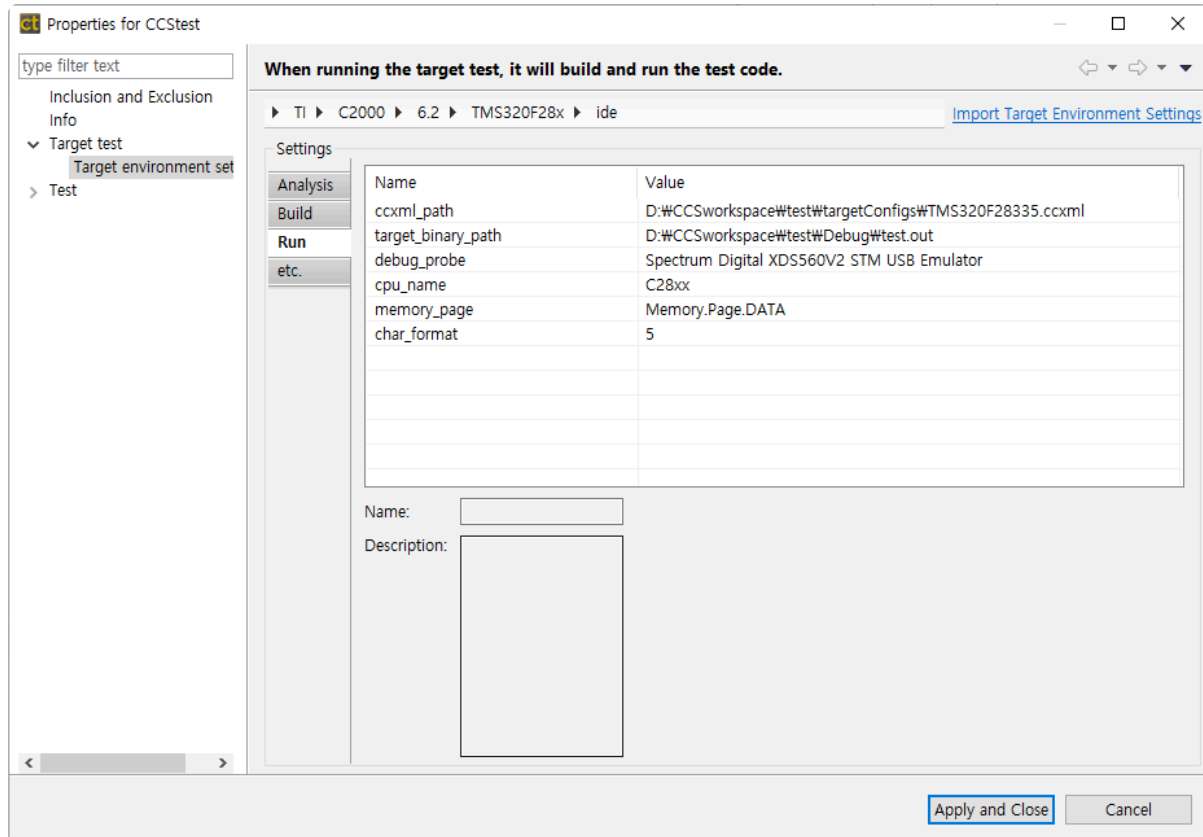
1. Create a CodeScroll Controller Tester project. For more information to create the project, refer to [Texas Instruments Code Composer Studio](#) in this document.
2. Right-click on the project in test navigator view and select [Properties] – [Target test] – [Target environment settings]. You can set up target environment in [Target environment settings]. Setting fields and the list of selectable debuggers depend on the toolchain selected for the project.
3. Select a debugger in [Target environment settings] of Controller Tester. This example selects IDE debugger to use Code Composer Studio debugger.
 ▶ TI ▶ C2000 ▶ 6.2 ▶ TMS320F28x ▶ ide
4. Enter needed informations on [Build] tab of [Target environment settings] for Code Composer Studio build. Following fields need to be filled and these are necessary.



- Fields of [Build] tab

ide_directory_path	Directory path of Code Composer Studio ex) C:\ti\ccs930
workspace	Directory path of Code Composer Studio workspace
project_name	Project name analyzed by Controller Tester

5. Enter needed informations on [Run] tab of [Target environment settings] for running target tests. Following fields need to be filled and these are necessary.



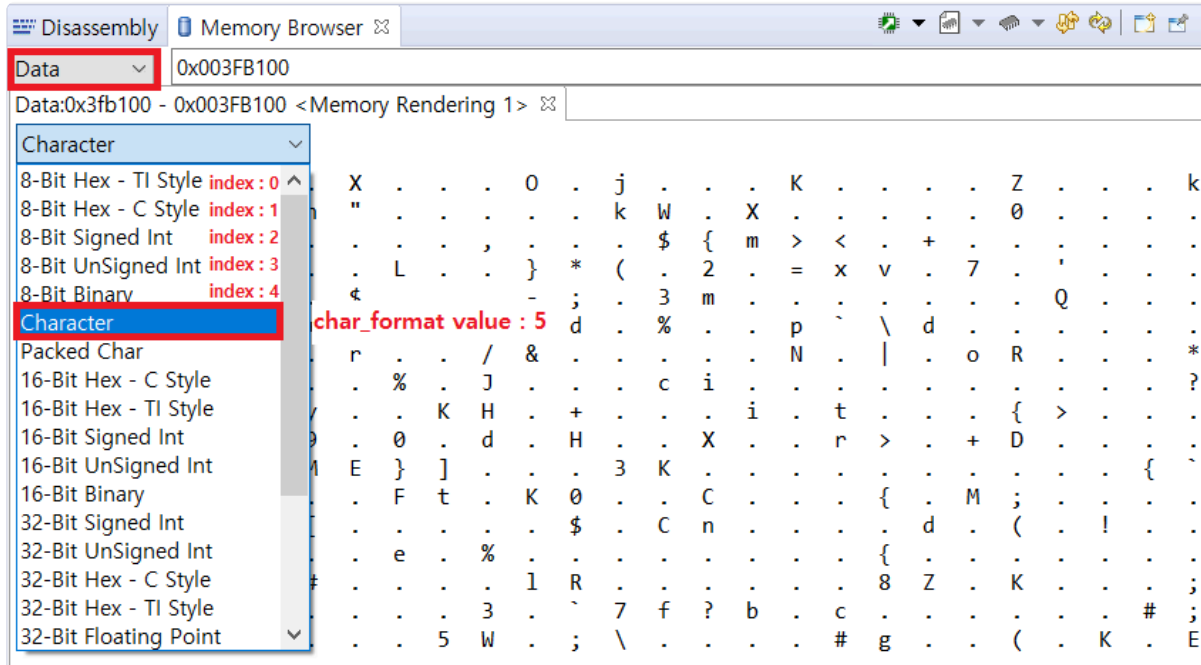
- Fields of [Run] tab

ccxml_path	Enter a path of Code Composer Studio target configuration file. Check the project path and target name. File name is the target name selected in Code Composer Studio ex) <i>project-path\targetConfig\target-name.ccxml</i>
script_path	Enter a path of execution script. You don't need to enter if you don't have the script.
target_binary_path	Enter a path of binary file created during build in Code Composer Studio. ex) <i>project-path\Debug\project-name.out</i>
debug_probe	Refer to front of '/' in [Device] of Code Composer Studio properties and enter a target device name. (<i>Spectrum Digital XDS560V2 STM USB Emulator</i> in example shown below)
cpu_name	Refer to back part of '/' in [Device] of Code Composer Studio properties and enter a target device name. (<i>C28xx</i> in example shown below)
memory_page	Enter a memory page that Code Composer Studio debugger can access (Refer to a Code Composer Studio Memory Browser image below)
char_format	Enter a order of <i>character</i> of Code Composer Studio data format. If <i>character</i> is the 6th from the top of the bar, enter 5 in the <i>char_format</i> value.

- Code Composer Studio properties
 - Right-click Code Composer Studio project and select [Properties] – [Debug] – [Device]



- Code Composer Studio Memory Browser



If Code Composer Studio is running during execution, compile error occurs.

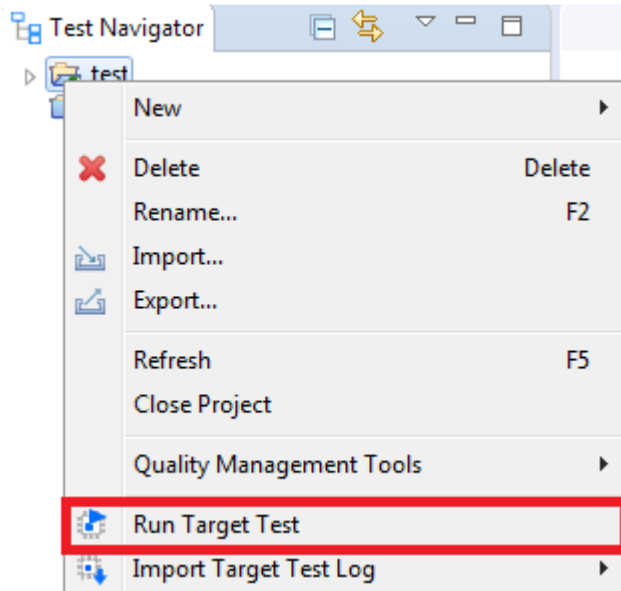
6. After finishing target environment settings, click [Finish] button. You are ready to do target tests.

2.5.3. Step3: Run the target test

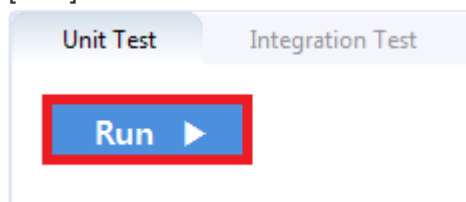
Before running the target test, you should stop using the workspace where the project you want to build is located. If you are using a workspace in the IDE, target testing does not work properly.

You can run a target test by selecting [Run Target Test] from the project context menu in the Test Navigator view or by clicking the [Run] button in the Test View.

- [Run Target Test]



- [Run]



✿ For more information on debug scripting in CCS, see the [Texas Instruments home page](#).

2.5.4. Debug the target test

1. After setting it as a target, right-click the test case in the 'Unit Test' view and click 'Check Debug Information'.
2. Run in debugging mode in Code Composer Studio.
3. Click [File] > [Open] File in Code Composer Studio.
4. Select the source file_number.c file with the function to be debugged in the Controller_Tester_workspace_path/.metadata/.plugins/com.codescroll.ut.embedded/project_name/TestFixture/cs
5. Add breakpoint where you want to debug
6. Run Debug

3. Target Build Guide

CodeScroll Controller Tester guides you through building target test code using target project information.

- [IAR Embedded Workbench IDE](#)
- [Texas Instruments Code Composer Studio](#)
- [CodeWarrior IDE](#)
- [Hightec Development Platform IDE](#)
- [Tasking VX IDE](#)
- [Renesas CS+ IDE](#)
- [MPLAB X IDE](#)
- [Microsoft Visual Studio](#)
- [GNU Compiler](#)

3.1. IAR Embedded Workbench IDE

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an IAR Embedded Workbench, enter the required information in the Analysis and Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Analysis tab

cpu	CPU of target that can be selected from Core of Processor variant
------------	---

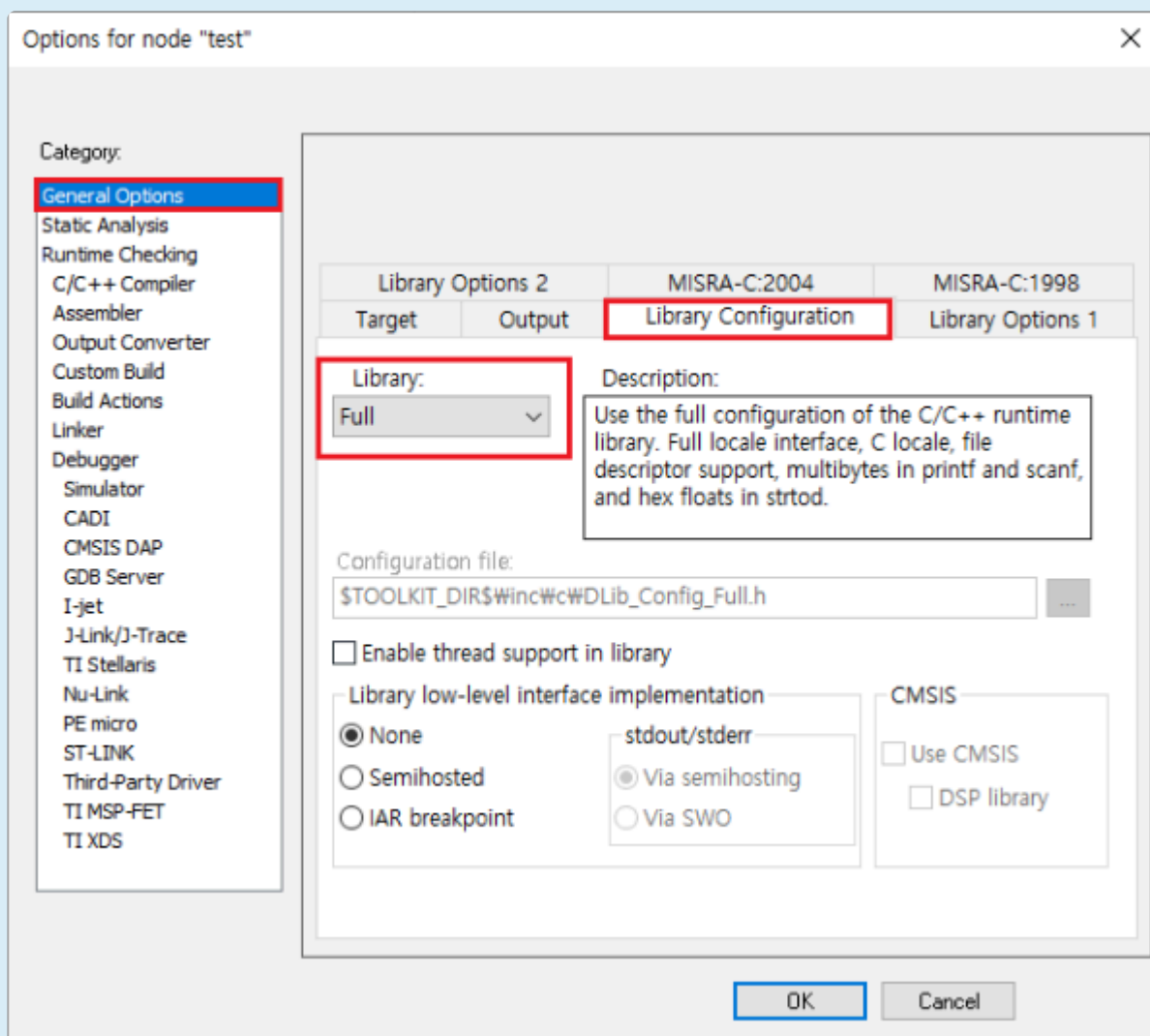
- Build tab

ide_directory_path	Installation path of the IAR Embedded Workbench IDE ex. <i>C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.4</i>
project_file_path	Project file (.ewp) path of IAR Embedded Workbench

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

✿ When using the IO function of `stdio.h`, it is necessary to change the library settings. Right click on the project in the workspace -> Options -> General Options -> Library Configuration -> Library tab and change it to Full.

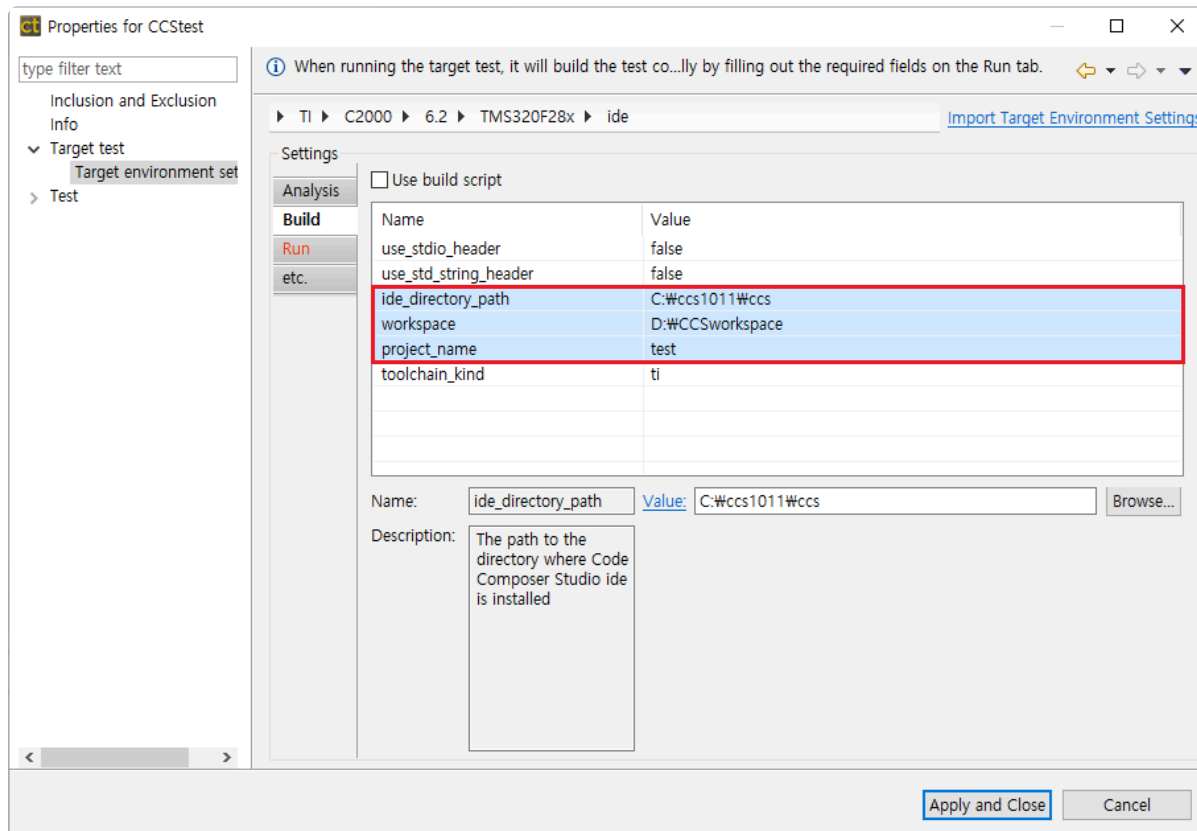


3.2. Texas Instruments Code Composer Studio

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an Code Composer Studio, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.



- Build tab

ide_directory_path	Directory path of Code Composer Studio ex.C:\ti\lccs930
workspace	Path to workspace directory in Code Composer Studio
project_name	The name of the Code Composer Studio project to be analyzed by Controller Tester

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.



If Code Composer Studio is running during execution, a compile error occurs.

3.3. CodeWarrior IDE

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an CodeWarrior project, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

ide_directory_path	Path to CodeWarrior IDE ex. <i>C:\Program Files (x86)\Freescale\CW for MPC55xx and MPC56xx 2.10, C:\Freescale\CW MCU</i>
ide_version	Classic or Eclipse(for MCUs)
project_file_path	In the case of Classic, the .mcp file named when creating the project, and in Eclipse, the .project file created when creating the project.

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

3.4. Hightec Development Platform IDE

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an Hightec IDE project, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

ide_directory_path	Path to Hightec IDE <i>ex. C:\HIGHTEC\toolchains\arm\v4.6.5.0</i>
project_directory_path	Path of project directory created by HighTec IDE

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

3.5. Tasking VX IDE

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an Tasking VX IDE project, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

ide_version	Version of Tasking VX IDE
makefile_path	Path of makefile created in Tasking VX IDE project
ide_directory_path	Path to the directory where Tasking VX IDE is installed ex. <i>C:\Program Files (x86)\TASKING\C166-VX v3.1r2</i>

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

3.6. Renesas CS+ IDE

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an Renesas CS+ IDE project, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

ide_directory_path	Directory path of Renesas CS + IDE <i>ex. C:\Program Files (x86)\Renesas Electronics</i>
ide_kind	IDE kind(CS+)
workspace_path	This is only necessary for the Renesas HEW IDE, so you do not need to enter it in CS +.
project_file_path	Project file path created by Renesas CS+(.mtpj)

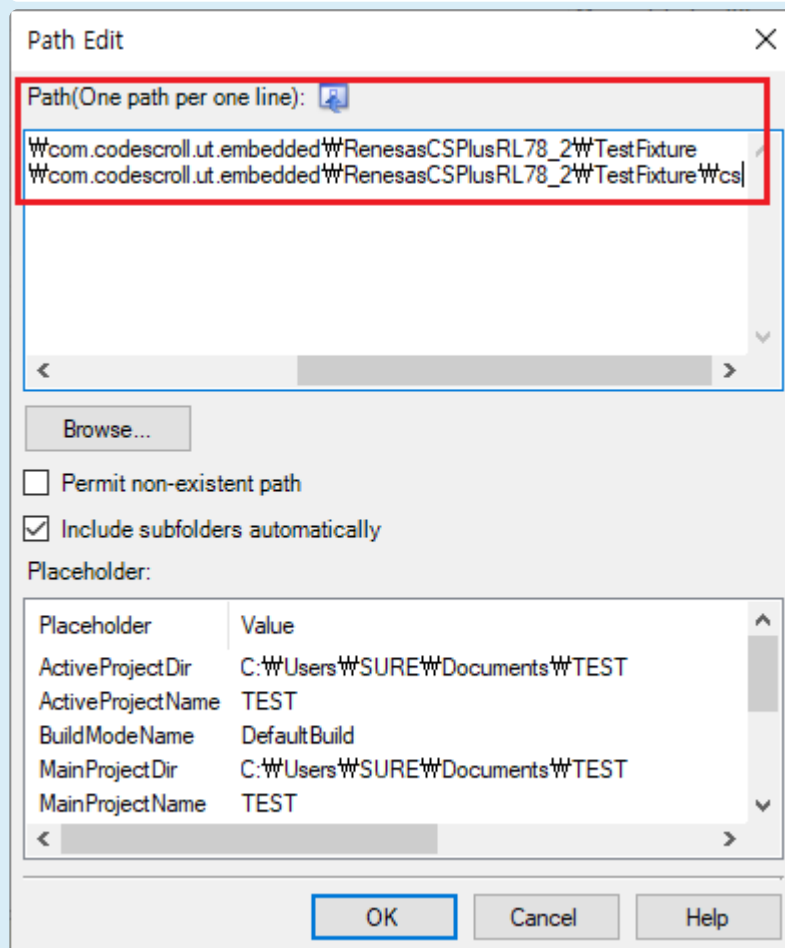
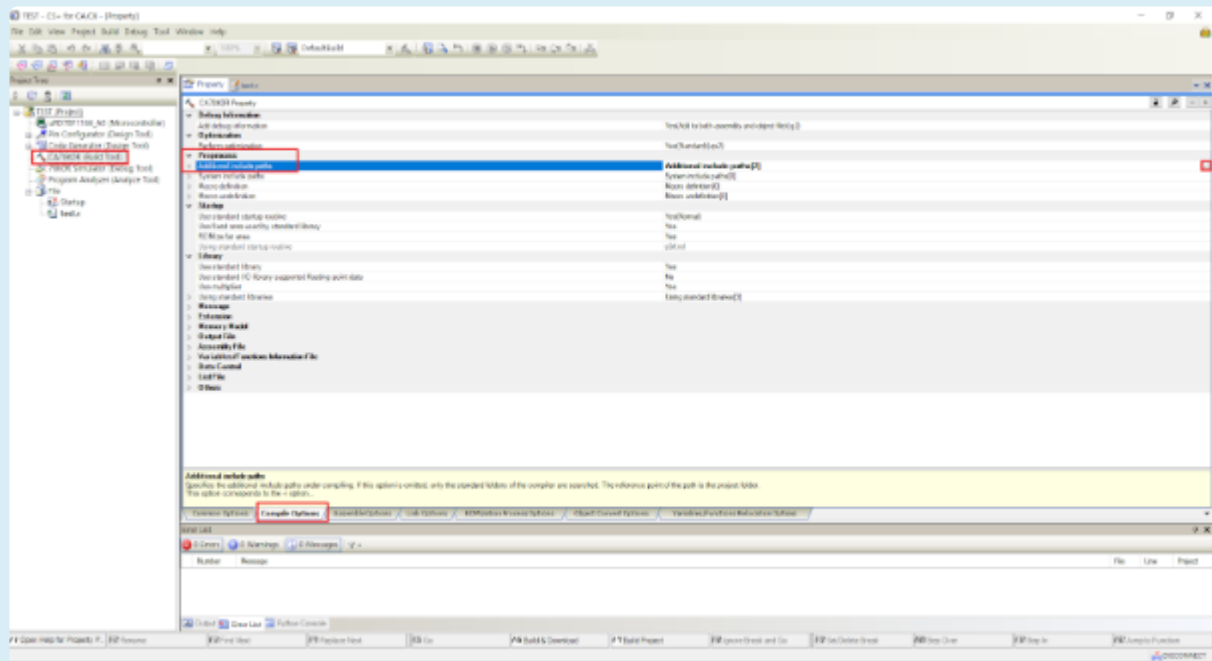
If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.



When exporting test codes from Controller Tester, some reference relative paths. To reference this path when building in the Renesas CS+, add an environment variable.

- Add below paths at property of Build Tool-> Compile Options -> Preprocess -> Additional include paths
- (CTWORKSPACE) \.metadata\plugins\com.codescroll.ut.embedded\ *CT project name* \TestFixture
- (CTWORKSPACE) \.metadata\plugins\com.codescroll.ut.embedded\ *CT project name* \TestFixture\cs



3.7. MPLAB X IDE

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an MPLAB X IDE project, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

ide_directory_path	Installation path of MPLAB X IDE ex. <i>C:\Program Files (x86)\Microchip\MPLABX\v5.35</i>
project_directory_path	Project directory path created in MPLAB X IDE

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

3.8. Microsoft Visual Studio

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an Microsoft Visual Studio project, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

ide_directory_path	Installation path of Microsoft Visual Studio ex. <i>C:\Program Files (x86)\Microsoft Visual Studio 10.0</i>	
build_configuration	Configuration and platform to test the target solution ex. Release	Win32
sin_path	File path of target solution (.sin file)	

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

3.9. GNU Compiler

The target preference page is automatically filled in according to the tool chain selected by the user. The type of debugger you can choose depends on your toolchain analysis settings.

To build an GNU Compiler code, enter the required information in the Build tab of the target preferences and click Done.

The contents to be filled out are as shown in the table below, which is mandatory.

- Build tab

makefile_path	Path of user-made makefile
----------------------	----------------------------

If you click the Done button on the target preference page without writing all the contents or if the path has been changed, you can set it again at 'Right click on the project in the test navigator-> Properties-> Test target-> Target environment'.

After setting the target environment and clicking the Run button in the unit test view, the controller tester builds the target test code.

4. Sharing Projects with Other Users

You can share the CodeScroll Controller Tester projects with others.

Controller Tester 3.3 or later uses [Export Project] and [Import Project] functions.

- [Guide to Share Projects](#)
- [Guide to Share RTV Projects](#)

4.1. (Ver.3.3 or later) Guide to Share Projects

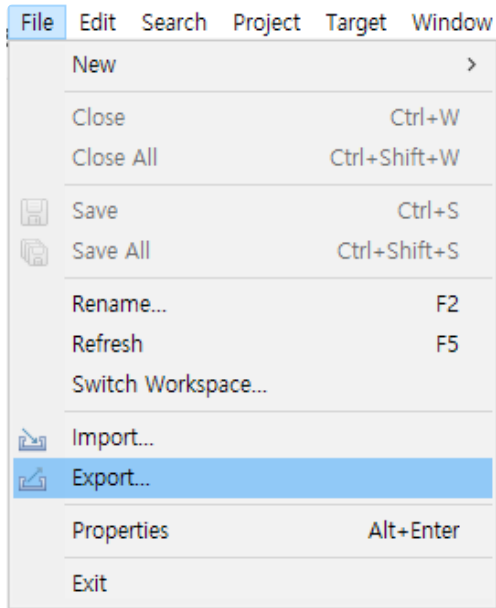
From Controller Tester 3.3, you can easily share a project with the [Export Project] and [Import Project] functions.

- [Export project](#)
- [Import project](#)

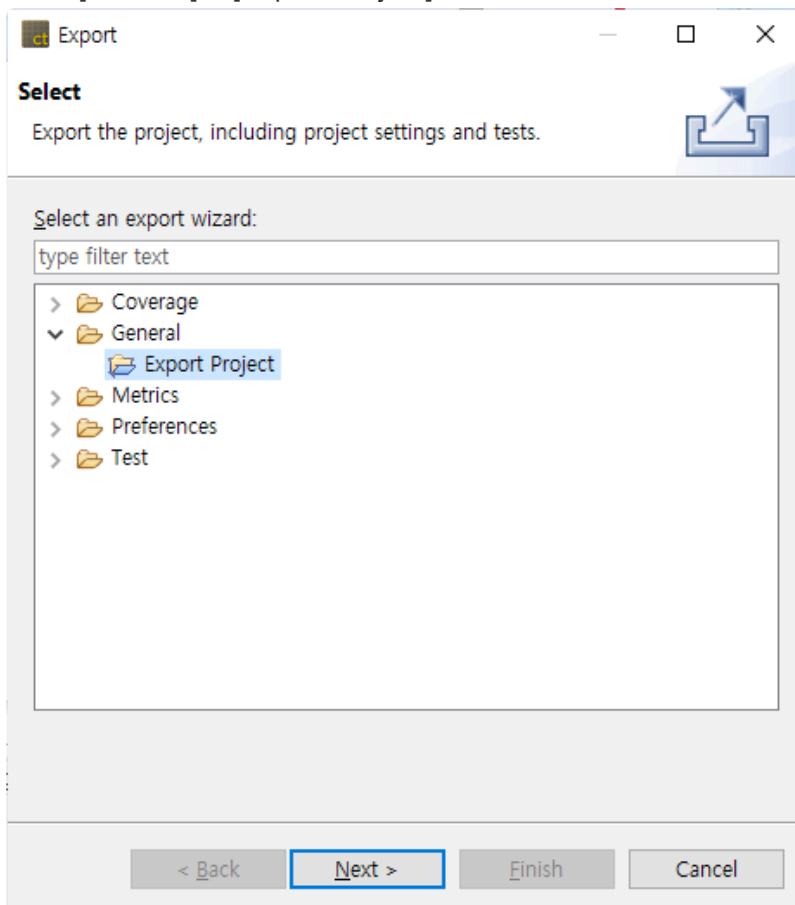
4.1.1. Export project

You can export projects, including project setup and testing.

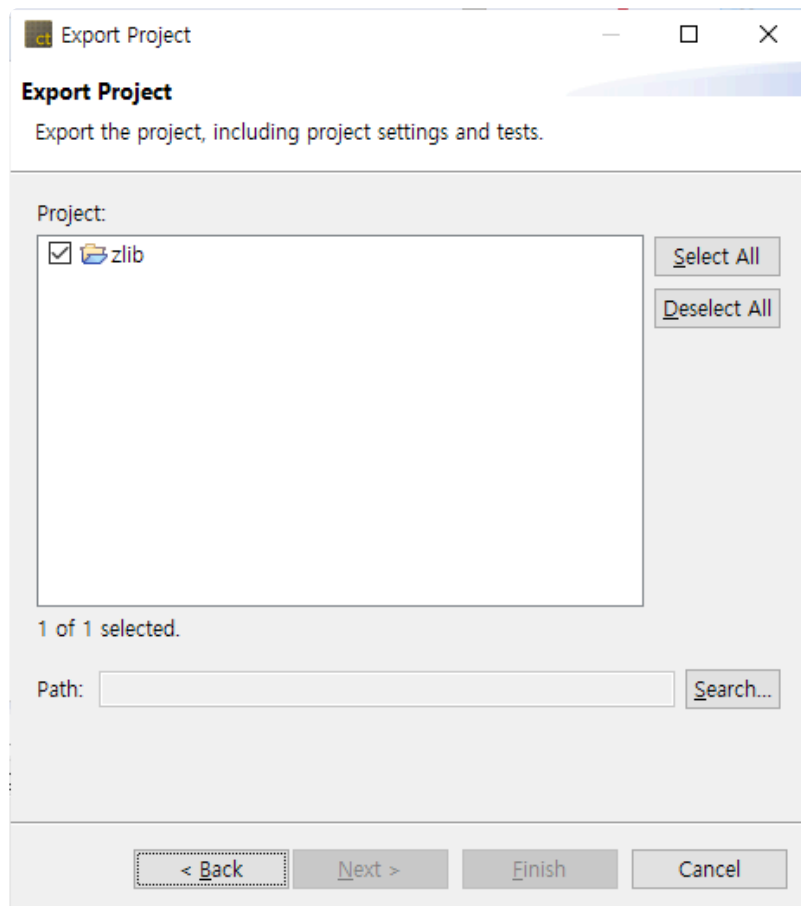
1. On the main menu, click [File] > [Export]. The Export Wizard opens.



2. Click [General] > [Export Project].



3. After selecting the project to export and the path to export, click the [Finish] button.



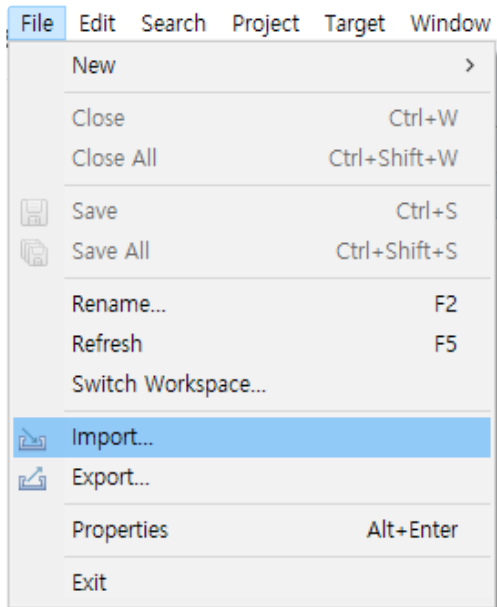
4. You can see that there is a folder containing the exported project name in the exported path. Compress the folder and move it to the computer of the user you want to share.

4.1.2. Import project

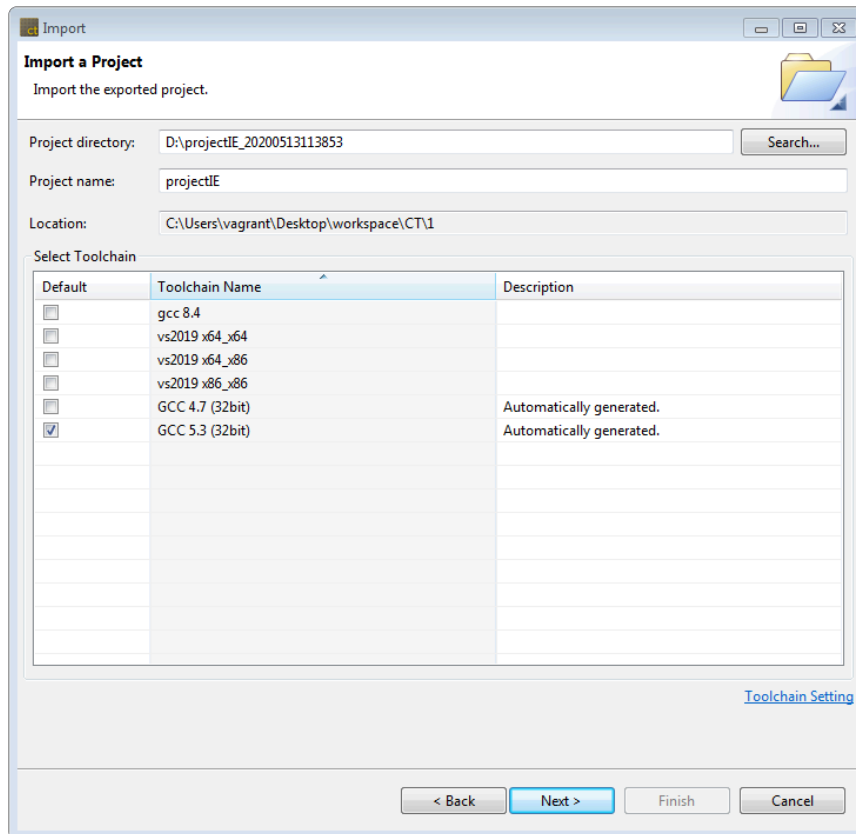
Using the Import Project function, you can import a project exported from another PC into the workspace.

Import general C/C++ Project

1. Click [File] > [Import] in the main menu. The Import Wizard opens.



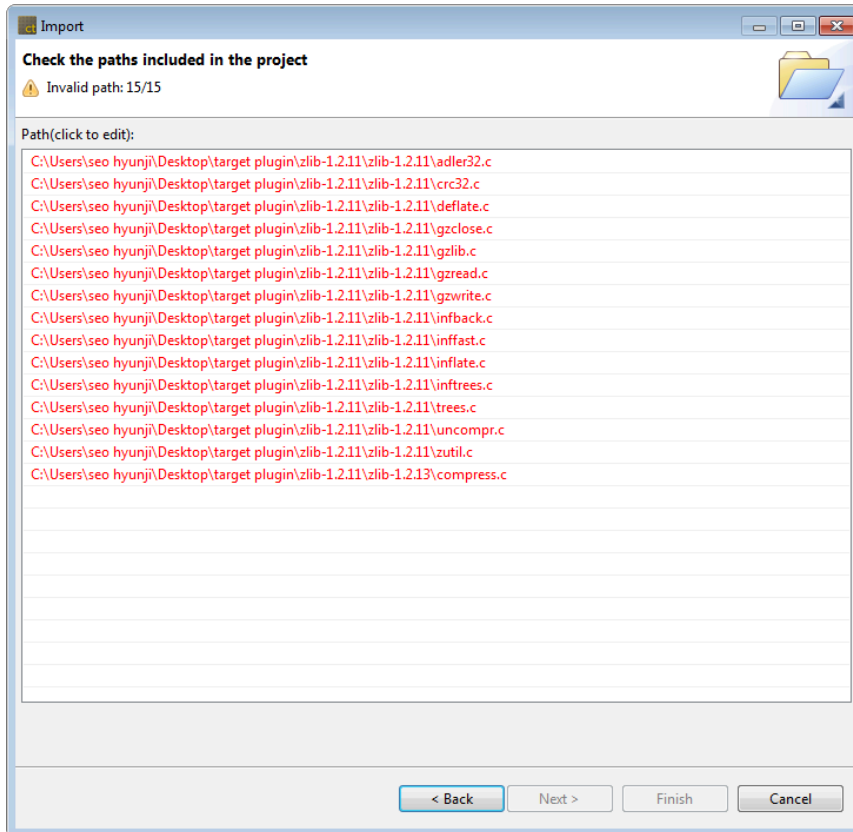
2. Click [General] > [Import Project] and then click the [Next] button.
3. Click the [Browse] button to find the directory corresponding to the exported project.
4. When you select a directory, the toolchain is automatically selected from the project information to be imported. If a project with the same name already exists in the workspace, you need to modify the project name.



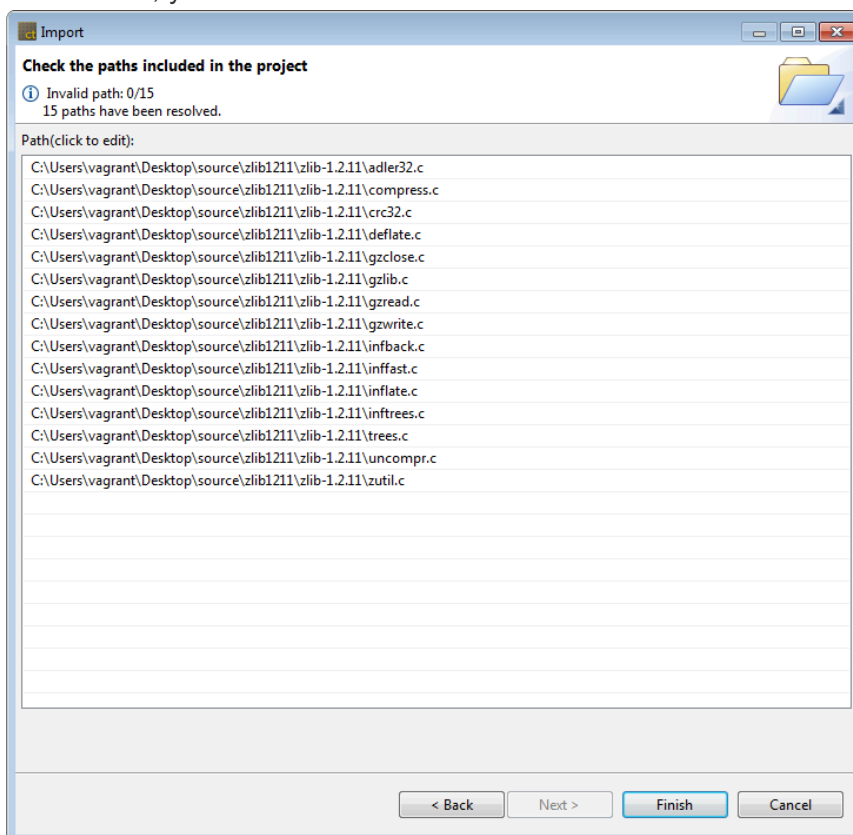
✿ If there is no toolchain with the same name as the toolchain of the project to be imported, you must first export and then import the toolchain of the project to be imported. For details, see [Import Toolchain] and [Export Toolchain] in the CodeScroll Controller Tester document.

#_Click the [Next] button.

1. You can check the source path included in the project to be imported. Invalid paths are marked in red and can be modified by clicking on the path window.



2. If there is an invalid path, modifying one file path automatically modifies the associated file path. At this time, you can check the number of modified routes at the top.



p(banner tip). If it is not in absolute path Windows format, the path is not checked for validity.

3. Click the [Finish] button.

Import RTV projects

RTV C/C++ projects can be imported in the same way as regular C/C++ project imports.

1. Click [File] -> [Import] in the main menu. In the Import Wizard, select [General] -> [Import Project] and click [Next].
2. Click the [Browse] button to select the directory of the project to be imported. When you select a directory, the toolchain is automatically selected from the project information to be imported. Click the [Next] button.



If there is no RTV server and toolchain information identical to the project to be imported, RTV server and toolchain information is automatically generated from the project to be imported.

3. You can check the source path included in the project to be imported. Invalid paths are marked in red and can be modified by clicking on the path list.
4. Click the [Finish] button.

Import target project

When importing a target C/C++ project, additional target preferences must be created.

1. Click [File] -> [Import] in the main menu. In the Import Wizard, select [General] -> [Import Project] and click [Next].
2. Click the [Browse] button to select the directory of the project to be imported. When you select a directory, the toolchain is automatically selected from the project information to be imported. Click the [Next] button.
3. In the case of a target project, the [Target Environment setting] window appears. The target environment setting is loaded from the project information to be imported. Items with invalid paths are displayed in red.

Import

Target Environment Settings

When running the target test, it will overwrite the source code with the test code. You can run the target test manually, or automatically by filling out the required fields on the Build/Run tab.

GNU Compilers > gcc > 5.3 > others > nodebugger [Import Target Environment Settings](#)

Settings

Analysis
Build
Run
etc.

Name	Value
language	c
Toolchain Name	GCC 5.3 (32bit)
Status	This toolchain is supported.
C Compiler	C:\gcc\5.3.0\32bit\bin\gcc.exe
System Header(C Compiler)	c:\gcc\5.3.0\32bit\bin\..\lib/gcc/mingw32/5.3.0/include;c:\gcc\5.3.0\32bi...
Library(C Compiler)	
C++ Compiler	C:\gcc\5.3.0\32bit\bin\g++.exe
System Header(C++ Compiler)	c:\gcc\5.3.0\32bit\bin\..\lib/gcc/mingw32/5.3.0/include/c++;c:\gcc\5.3.0...
Library(C++ Compiler)	

Name:

Description:

< Back Next > Finish Cancel

✿ Even if it is not a target C/C++ project, if it is a project that includes target environment settings, the target environment setting window appears when [Import Project] is executed.

! Even if it contains an invalid path, you can complete the target environment setup and proceed to the next one, but the one-click target test may not be executed.

4. Complete the target environment settings and click the [Next] button.
5. You can check the source path included in the project to be imported. Invalid paths are marked in red and can be modified by clicking on the path list.
6. Click the [Finish] button.

4.2. (Ver.3.2 or earlier) Guide to Share RTV Projects

RTV projects can be easily shared because the toolchain and source file information can be fetched from the RTV server.

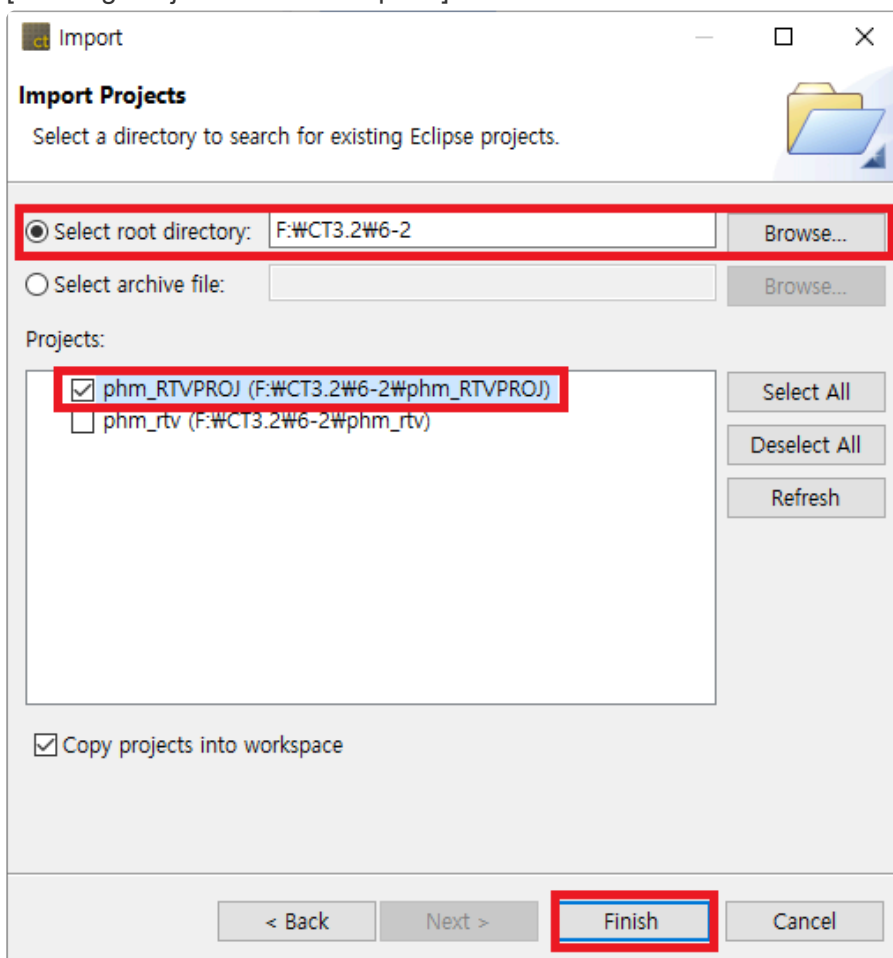
The step-by-step scenario according to the usage environment is as follows, and the RTV project can be shared when the scenario is followed.

- [Project sharing scenario](#)
- [RTV server user guide](#)

4.2.1. Project sharing scenario

When using the [Existing Projects into Workspace] function

1. When you create a RTV project, a RTV project directory (hereinafter referred to as RTV_A project) is created under the Controller Tester workspace.
2. The user who wants to share the project receives the RTV_A project directory created in the above step, and copies and pastes the RTV_A project directory into the Controller Tester workspace path that he uses.
3. Select top-level path to the project directory to import the projects using [Import] > [General] > [Existing Projects into Workspace] function.



4. Information required for the project is received from the RTV server, and [the toolchain or resource setting of the project is incorrect. If you want to reset automatically?], Click 'Yes' to complete the RTV setup (RTV server and toolchain registration used when creating the project).
5. You can see that an RTV project (hereinafter RTV_A' project) with the same name as RTV_A has been created in the Controller Tester test navigator view.
6. Right-click the RTV_A' project in [Test Navigator View] and perform [Reanalysis].
7. This should be done when connected to the same RTV server.

When using the [C/C++ Project from RTV Build] function

1. When you create an RTV project, an RTV project directory (hereinafter RTV_A project) is created under the Controller Tester workspace.
2. The user who wants to share the project connects to the same RTV server where the above project was created from Controller Tester that he uses, and registers the same RTV toolchain.
3. In the project creation wizard, select [C/C++ Project from RTV Build] to create an RTV project (hereinafter RTV_A' project).
4. Import the `$(project folder)/.csdata/link.mk` file from the RTV_A project folder in the Controller Tester workspace and overwrite the link.mk file in the RTV_A' project folder.
5. If you want to share the same test data, check the below.

* If the path where the source files are located is long, the entire source file may not be imported properly. If the path where the source files are located is too long, make sure to specify the CT's global path just below the drive. (ex. `C:\temp`)
To modify the CT global path, open the CodeScroll.ini file in the location where the CT package is installed and replace the default under the -g option with the new global path to set.

```
1 -startup
2 plugins/org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar
3 --launcher.library
4 plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.500.v20170531-1133
5 -data
6 @noDefault
7 -g
8 C:\temp
9 -vmargs
```

* Even if you share the same project, coverage results may differ if you create each unit test. When you share a test, you must export the test using the [Export] > [Export Test] feature, and then import the test you exported using the [Import] > [Import Test] feature.

Export tests
Export project's test informations.

Export path: *\$(temp_folder)*

Unit Test
☒ Test
☒ Test Data

Integration Test
☒ Test
☒ Test Data

Stub
☐ Connected Stub
☒ All Stub

Option
☐ Overwrite existing test files without warning
☐ Export only checked tests in Unit/Integration Test View.

Import tests
Import project's test informations.

Import path: *\$(temp_folder)*

Unit Test
☒ Test
☒ Test Data

Integration Test
☒ Test
☒ Test Data

Stub
☐ Connected Stub
☒ All Stub

Option
☐ Overwrite existing test files without warning
☐ If the same stub exists already, it is not imported.
☒ If the same stub exists, it is added as a new stub.

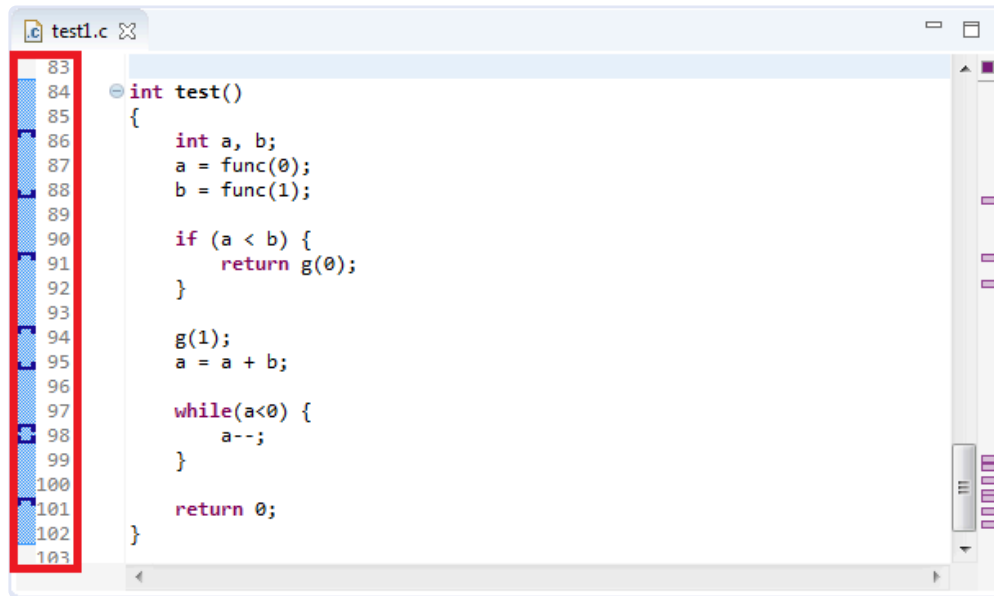
4.2.2. RTV server user guide

When using one RTV server

1. When RTV server has a project built using the csbuild capture function
 - a. Projects can be imported according to the project sharing scenario above, without the need for additional settings.
2. When the RTV server is connected, but the server (IP/Port) information is different
 - a. Since server (IP/Port) information at the time of project creation is imported, existing server information is imported and toolchain information is not imported.
 - b. After modifying the server information to access the existing server, import the toolchain with the same name by importing the toolchain. At this time, the path of the tool chain used in the project should be the same.

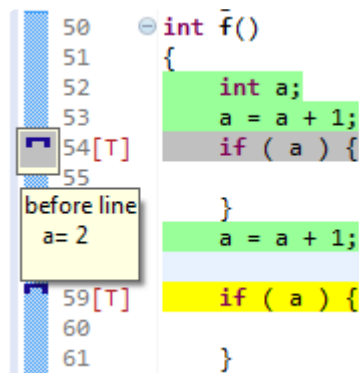
! Sharing of RTV projects can be difficult if you are using more than one RTV server (same source file, tool chain, or if you want to receive and use a virtual machine file with RTV server installed).

The List of variable/expression represents the variable/expression values executed with the test case. The list of variables/expressions added to the entire source code can be checked in [List of Variable/Expression] in the toolbar menu of [Debug Information View].



You can also check the variable/expression information to debug in the marker in the source code editor. When you add a variable/expression to debug, the additional position is expressed as a marker in the source code editor, and when you mouse over each marker, you can see the list of variables/expressions added at that position.

If you select a test case that contains debug information, each marker displays the result of the variable/expression executed by the test case.



The stack trace and the executed variable/expression value can be used to identify the cause of the error in the test case that executed [Inspect Debug Info].



For more information on adding variables/expressions to debug, see [Add Variables/Expressions to Debug] in the CodeScroll Controller Tester document.

6. (After Ver.3.4) Source Code Modification and Test Reconfiguration

After designing tests source code can be modified. CodeScroll Controller Tester offers [Test reconfiguration] feature to detect source code modifications and help reconfiguring tests affected by the modification.



Reflect modified source codes using [Refresh RTV Source File] feature before using [Test reconfiguration] in case of RTV projects and RTV target projects.

Controller Tester divide source code modifications into three cases.

- Modifying names of test or stub functions.
- Modifying names of global variables used in tests.
- Modifying names or the numbers of return type or parameter of test functions.

In cases of detectable modification by Controller Tester, refer to [In Cases of Detected Modification by Controller Tester](#) and in cases of undetectable modification by Controller Tester, refer to [In Cases of Undetected Modification by Controller Tester](#).

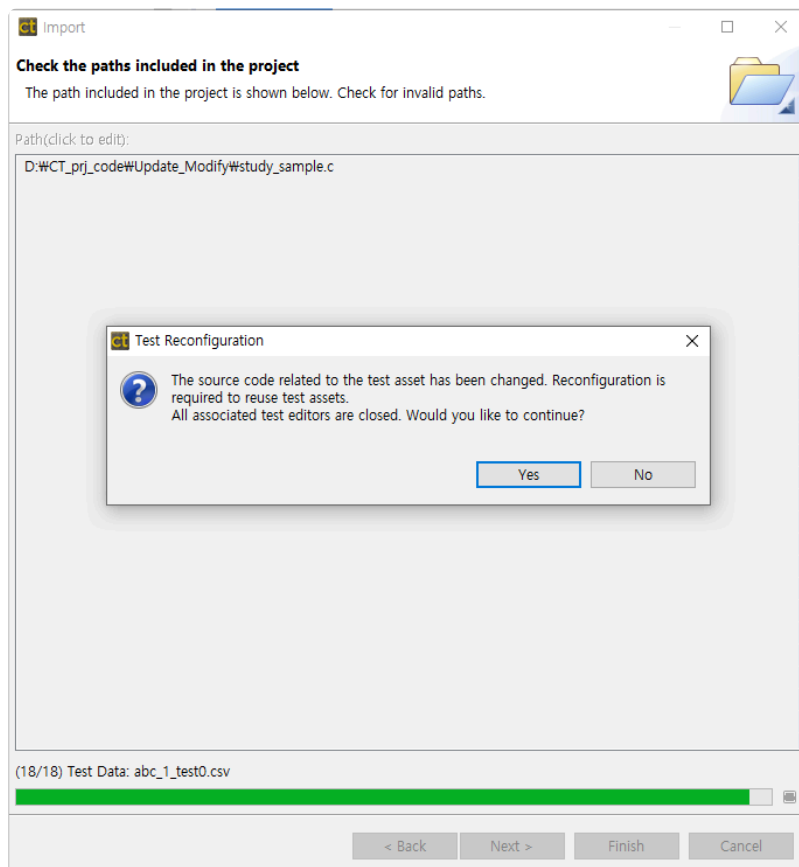
6.1. Run [Test Reconfiguration]

How to automatically use [Test Reconfiguration] feature

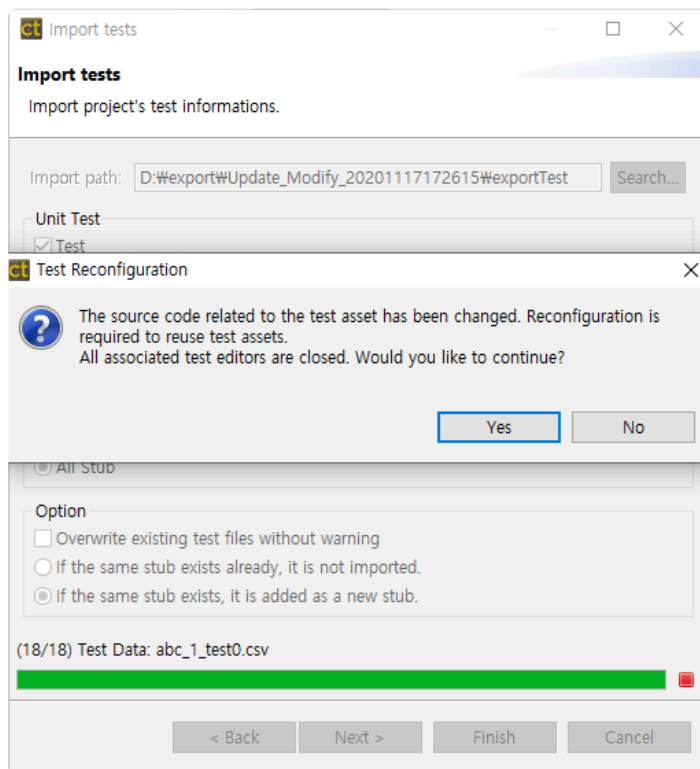
Controller Tester automatically detects following modifications.

- When differ present project information from imported project information using [Import Project] feature.
- When differ present project information from imported test information using [Import test] feature.
- When detect source code modification after analyzing project.

When differ present project information from imported project information using [Import Project] feature



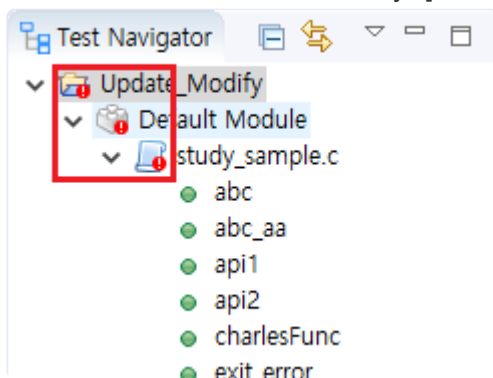
When differ present project information from imported test information using [Import test] feature



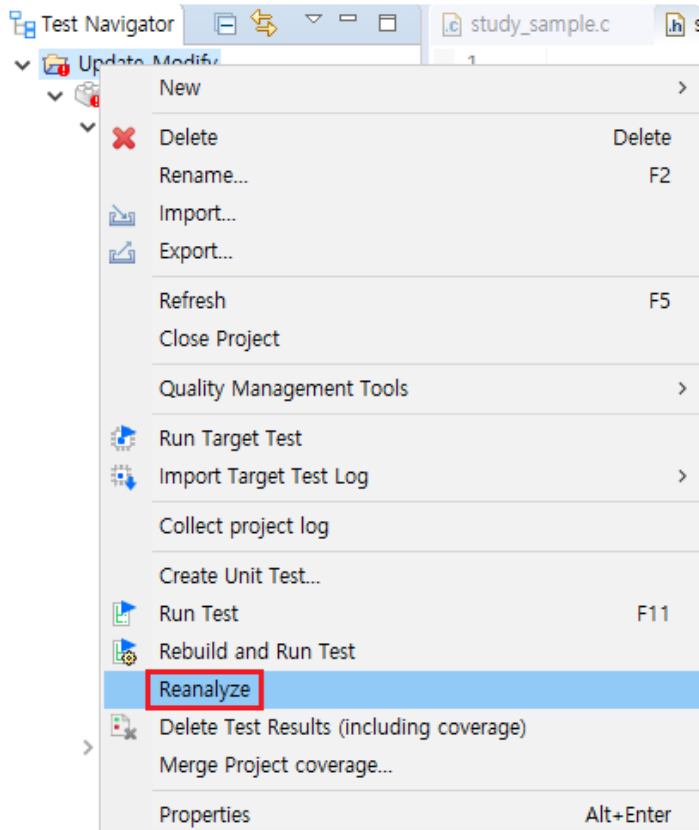
When detect source code modification after analyzing project

You can use [Test Reconfiguration] feature when Controller Tester detects source code modification after project analysis or reanalysis.

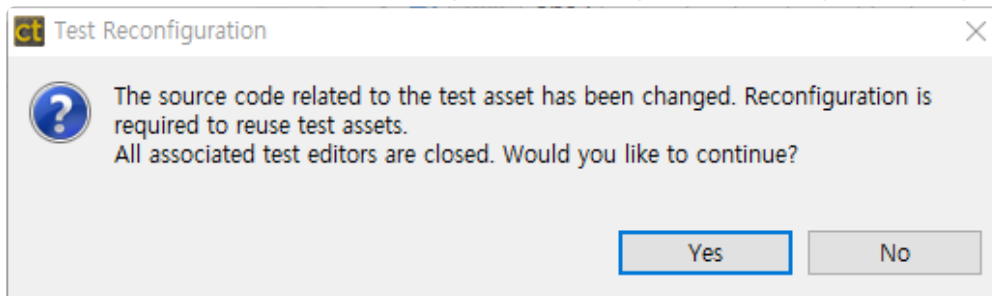
1. When the source codes modify, [Test Navigator] view indicates whether the change was made.



2. Select [Reanalyze] in project context menu or run tests to analyze the source codes.



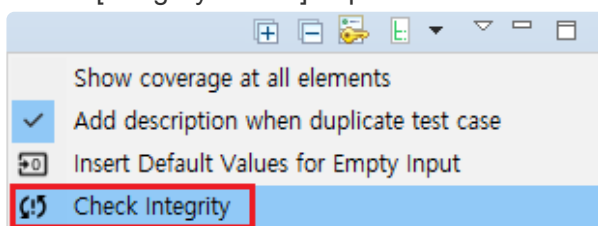
- Click [Yes] button in [Test Reconfiguration] dialog, then a dialog for reconfiguration appears.



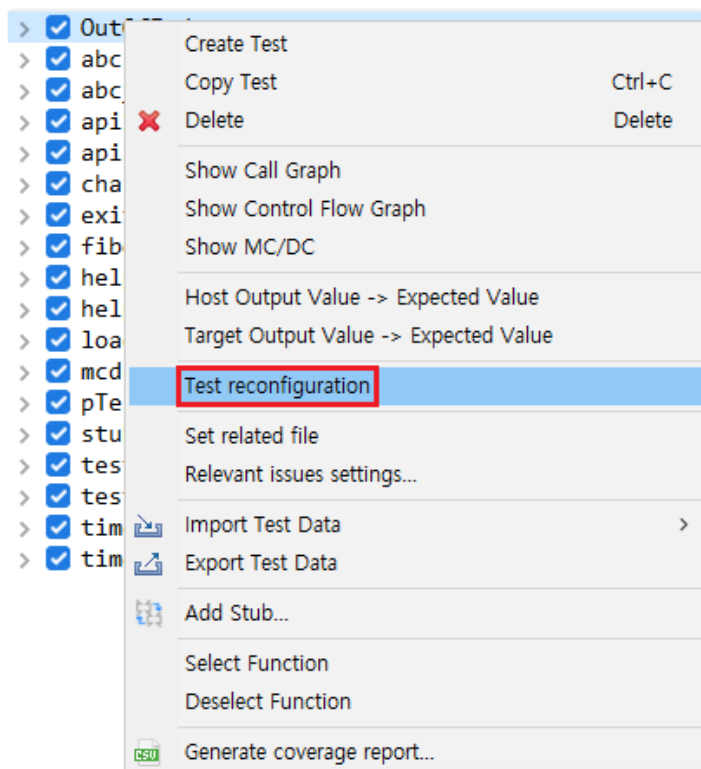
How to manually use [Test Reconfiguration] feature

If you click [No] button in [Test Reconfiguration] dialog or [Cancel] button while reconfiguration, following two method allow to use [Test Reconfiguration] feature.

- Select [Integrity Check] in pull-down menu of unit test view.



- Select [Test reconfiguration] to use [Test Reconfiguration] feature in function context menu or test context menu of unit test view.



You can design a new test based on original test using [Test reconfiguration] feature.

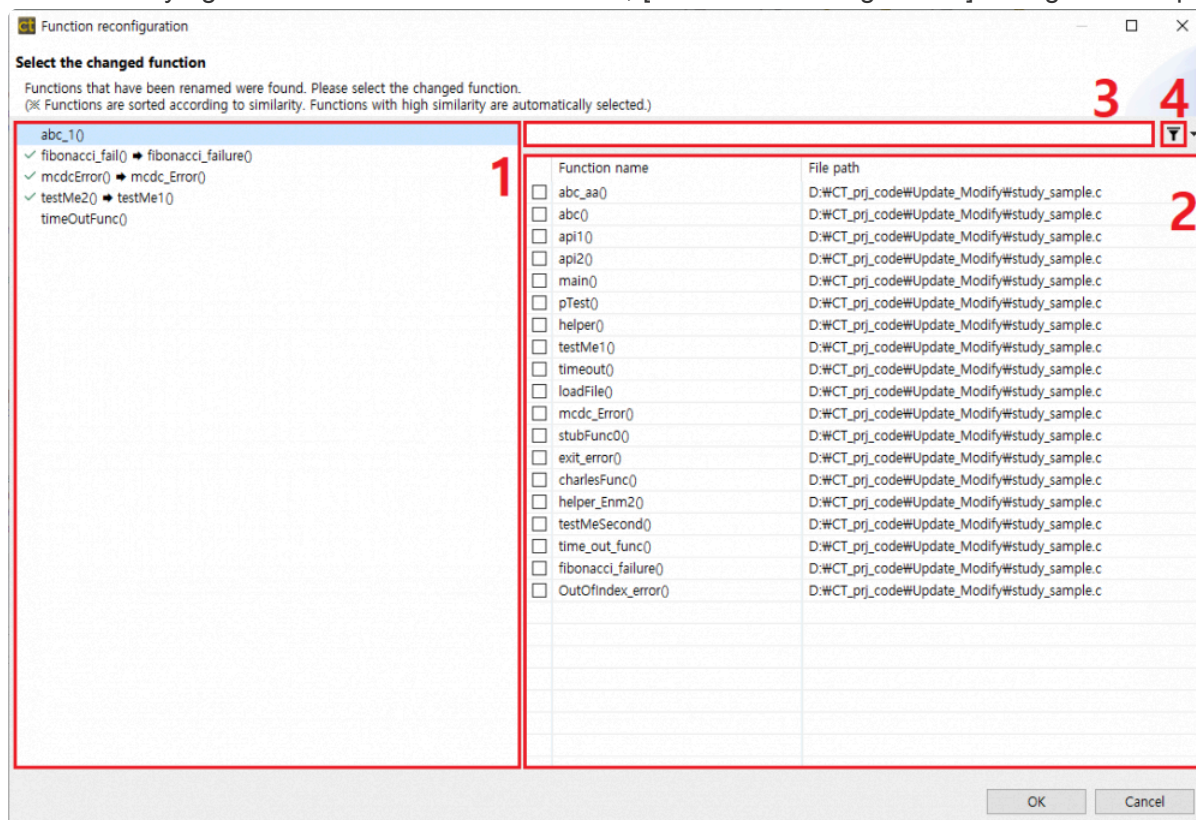
6.2. In Cases of Detected Modification Automatically

When re-analyze or run the tests after modifying source codes, Controller Tester detects modifications with integrity checker. Controller Tester divide source code modification with three cases.

- Modifying names of test or stub functions.
- Modifying names of global variables used in tests.
- Modifying name or number of return type or parameter of test functions.

Modifying names of test or stub functions

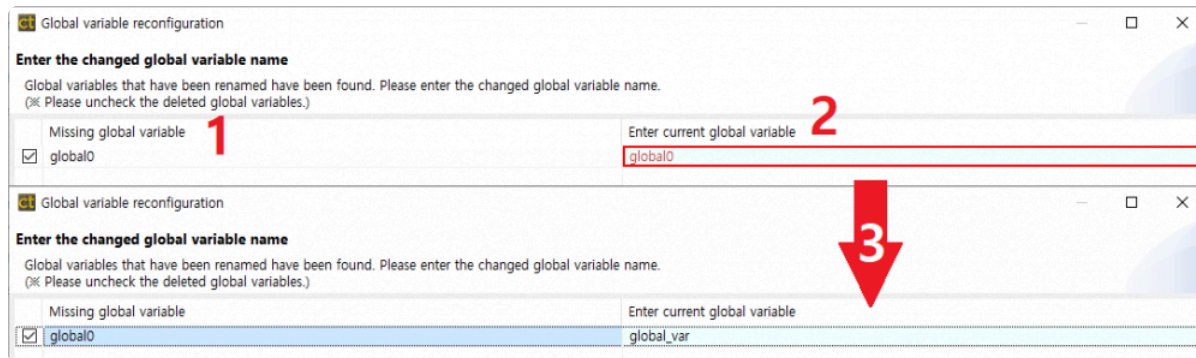
When modifying names of test or stub functions, [Function reconfiguration] dialog shows up.



1. Left area is a list of function that modification detected. Functions that finish reconfiguration are marked with ✓.
2. Left area is a list of function contained in present source code.
 - It's sorted by similarity of function name.
 - Function with high similarity is connected automatically.
3. It allow to search a function name. (*: any string, ?: any letter)
4. It show or hide functions with tests.

Modifying names of global variables used in tests

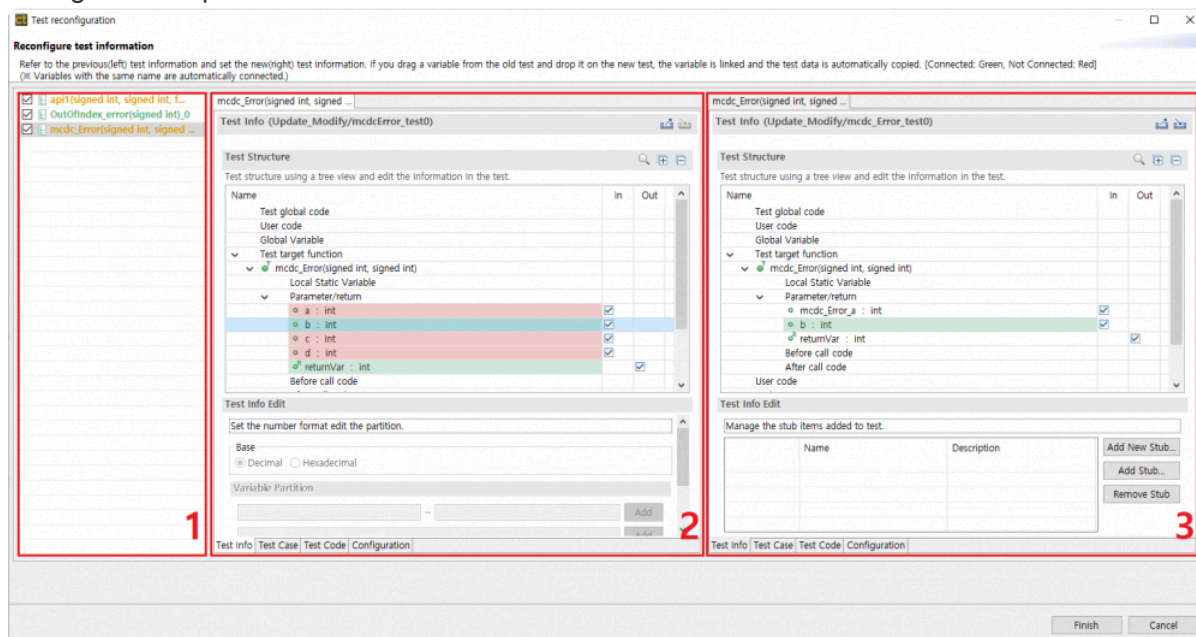
When modifying names of global variables used in tests, [Global variable reconfiguration] dialog shows up.



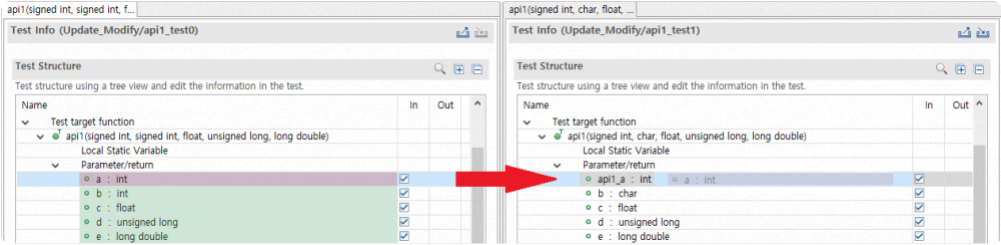
1. Left area is a list of global variables that cannot find.
 - Uncheck check boxes when variables are deleted.
2. Right area contain text boxes for entering present global variable.
 - When user modify a global variable name, it shows global variable list in order of similarity.
3. When user enter a valid variable, red mark in the text box disappear.

Modifying name or number of return type or parameter of test functions

When modifying name or number of return type or parameter of test functions, [Test reconfiguration] dialog shows up.



1. A list of modified functions.
2. Test information about function before modifying.
 - If a variable connect to test information after modifying, it's displayed in green and if not, it's displayed in red.
3. Test information about function after modifying.
 - When select a variable of function before modifying, it shows connected variable with selected variable.
 - When drag a variable of function before modifying and drop to a variable of function after modifying, test data are copied.



6.3. In Cases of Undetected Modification Automatically

Controller Tester cannot detect following types of modification with integrity check.

- Modify value type of global variable.
 - Test build error (when implicit type conversion is unable)
 - Test run error (runtime error including memory overflow, etc)
- Modify symbols excluding global variables.
 - Modify lower type of parameters, symbol added with macro by user, static variable, etc.
- Side effect by modifying function position
 - error that test cannot access to global variable
- Modify build stubs.

When modify value type of global variable, symbols excluding global variables, and function position, user reconfigures test using [Test reconfiguration] feature. When modify build stubs, user delete build stubs because build stubs are not target of integrity check.