# CT USER GUIDES

SOFTWARE FOR SAFE WORLD



🖸 СТ



2023.12 — Last update: Dec 11, 2023

Suresofttech

# **Table of Contents**

1.	Source Code Modification and Test Reconfiguration	6
	1.1. Run [Test Reconfiguration]	7
	1.2. In Cases of Detected Modification Automatically	13
	1.3. In Cases of Undetected Modification Automatically	19
2.	Collaboration Guide	20
	2.1. Team Testing Usage Guide	21
	2.1.1. Project Initialization	24
	2.1.2. Commit and Update	26
	2.1.3. Test Result Merge	34
	2.1.4. On/Offline Mode	36
	2.2. Sharing Projects with Other Users	38
	2.2.1. (Ver.3.3 or later) Guide to Share Projects	39
	2.2.1.1. Export project	40
	2.2.1.2. Import project	42
	2.2.2. (Ver.3.2 or earlier) Guide to Share RTV Projects	47
	2.2.2.1. Project sharing scenario	48
	2.2.2.2. RTV server user guide	51
	2.3. Guides to Import Coverages	52
	2.3.1. Import Coverages by Version	53
	2.3.2. Import Coverages by Conditional Operation Option	54
	2.3.3. Import Coverages by Coverage Type	55
3.	Scenario(Time-based) Test Usage Guide	56
	3.1. Check for changes in specific variables during a scenario test run	60
	3.2. Determine whether to call a function based on the value of the global variable	64
4.	C++ Test Guide	69
	4.1. Guides for C++ Test Using the Class Factory View	70
	4.1.1. Basic Concept for C++ Test	71
	4.1.2. Using the Object Creation Code of Abstract Class for Testing	72
	4.1.3. Design C++ Tests Using Class Factory	73
	4.1.4. Using Mock Objects in C++ Test	74
	4.1.4.1. Creating mock objects	75
	4.1.4.2. Generate specifications about mock objects	76
5.	CI/CD Environment and CLI Guide	79
	5.1. CT Jenkins plugin Usage Guide	80
	5.1.1. Creating Freestyle Project	82
	5.1.2. Creating Pipeline Project	85
	5.1.3. Check the result	89
	5.2. CLI Guide	92
	5.2.1. CLI Project Path Reset	93
6.	Test in Real Target Environments	95

6.1. Target Test Guides	96
6.1.1. Texas Instruments Code Composer Studio	97
6.1.2. STM32cubeIDE	99
6.1.3. Wind River Workbench	. 108
6.2. Debugger User Guides	. 115
6.2.1. Lauterbach TRACE32	. 116
6.2.1.1. Supported target list that can generate cmm script automatically	117
6.2.1.2. Step1: Setting target environment in CT	. 118
6.2.1.3. Step2: Run the target test	. 119
6.2.1.4. Debug the target test	
6.2.2. PLS Universal Debug Engine (UDE)	. 121
6.2.2.1. Step1: Create a workspace in UDE IDE	. 122
6.2.2.2. Step2: Setting target environment in CT	
6.2.2.3. Step3: Run the target test	. 125
6.2.2.4. Debug the target test	. 126
6.2.3. iSYSTEM winIDEA Debugger	. 127
6.2.3.1. Preparation for use of iSYSTEM winIDEA	. 128
6.2.3.2. Step1: Creating and setting up a winIDEA workspace	. 129
6.2.3.3. Step2: Setting target environment in CT	. 134
6.2.3.4. Step3: Run the target test	. 135
6.2.3.5. Debug the target test	. 136
6.2.4. IAR Embedded Workbench C-SPY Debugger	. 137
6.2.4.1. Step1: Creating an IAR embedded workbench project	
6.2.4.2. Step2: Setting an IAR project	
6.2.4.3. Step3: Setting target environment in CT	. 142
6.2.4.4. Step4: Run the target test	
6.2.4.5. Debug the target test	
6.2.5. Texas Instruments Code Composer Studio (CCS v4 and later)	
6.2.5.1. Step1: Create a project in Code Composer Studio	
6.2.5.2. Step2 : Setting target environment in CT	
6.2.5.3. Step3: Run the target test	. 151
6.2.5.4. Debug the target test	. 152
6.2.6. Microchip MPLAB IDE	
6.2.6.1. Step1: Debugger script settings	
6.2.6.2. Step2: Setting target environment in CT	
6.2.6.3. Step3: Run the target test	
6.3. Target Build Guide	
6.3.1. IAR Embedded Workbench IDE	
6.3.2. Texas Instruments Code Composer Studio	
6.3.3. CodeWarrior IDE	
6.3.4. Hightec Development Platform IDE	
6.3.5. Tasking VX IDE	
6.3.6. Renesas CS+ IDE	
6.3.7. MPLAB X IDE	
6.3.8. Microsoft Visual Studio	. 168

	6.3.9. GNU Compiler	169
7.	Identifying the Cause of a Test Error	170
8.	Virtual Address Usage Guide	172
9.	Navigate Source Codes	176

### 1. Source Code Modification and Test Reconfiguration

After designing tests source code can be modified. CT 2023.12 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.

CT 2023.12 divide source code modifications into four cases.

- Modifying names of classes used in tests.
- Modifying names of test or stub functions.
- Modifying names or type of global variables used in tests.
- Modifying names of member functions used in class codes.
- Modifying names or the numbers of return type or parameter of test functions.
- Modifying the code of the target function to be injected with the fault.

In cases of detectable modification by Controller Tester, refer to <u>In Cases of Detected Modification</u> <u>Automatically</u> and in cases of undetectable modification by CT 2023.12, refer to <u>In Cases of Undetected</u> <u>Modification Automatically</u>.

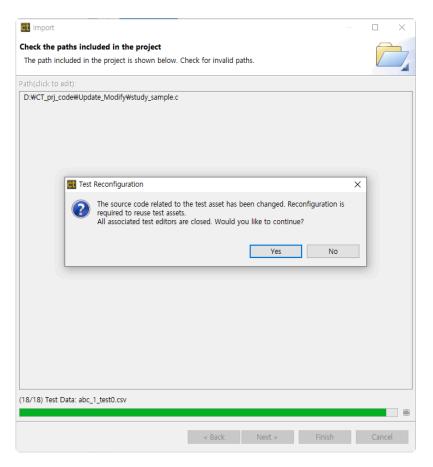
# 1.1. Run [Test Reconfiguration]

### How to automatically use [Test Reconfiguration] feature

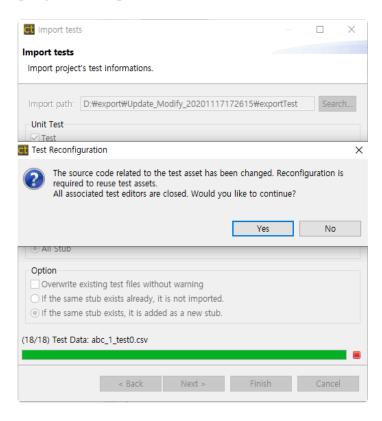
CT 2023.12 automatically detects following modifications.

- 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 analyze the project after writing the fault injection code in a location where fault injection is not possible.

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



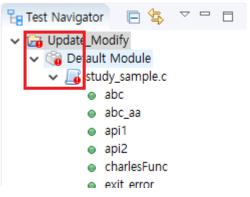
# When differ present project information from imported test information using [Import test] feature



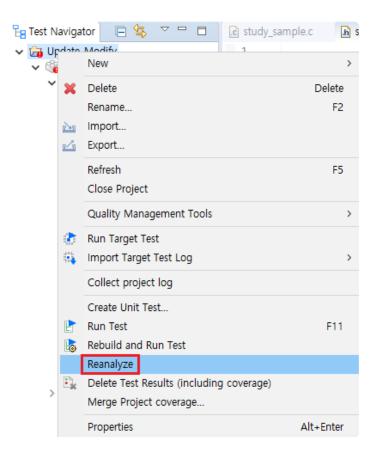
#### When detect source code modification after analyzing project

You can use [Test Reconfiguration] feature when CT 2023.12 detects source code modification after project analysis or reanalysis.

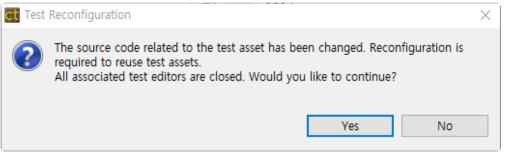
1. When the source codes modify, the Test Navigator View indicates whether the change was made.



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



3. Click [Yes] button in [Test Reconfiguration] dialog, then a dialog for recofiguration appears.



# When analyze the project after writing the fault injection code in a location where fault injection is not possible

When reanalyzing the project, if there is fault injection information that satisfies the condition below, the Reconfiguring Fault Injection dialog appears.

• When activate a line in a location where the fault cannot be injected and write fault injection code.

In the Reconfiguring Fault Injection dialog, you can see where faults cannot be injected and fault injection information.

e main()	type filter text	type filter text	
		<ul> <li>13:</li> </ul>	
	✓ <u>14</u> : while ( b < 5) {	✓ <u>14</u> : while (b < 5) {	
	✓ <u>15</u> : a++;	✓ <u>15</u> : a++;	
	16: }	16: }	
	17:	17:	
	✓ <u>18</u> : a = 19: b + 5;	<u>18</u> : a = 19: b + 5;	
	20:	20:	
	21: for (int i = 0; i < 10; i++)	20. 21: for (int i = 0; i < 10; i++)	
		✓ 22: {	
	Inserted Before: 1 /* test */	Inserted Before: 1/* test */	
	1/* test */	1 /* test */	
	<	> <	
		Inserted After:	
	(Failed) Inserted After:		
	<pre>1 int testVar = 0;</pre>	^ <b>1</b>	

The Reconfiguring Fault Injection dialog allows you to reuse fault injection information previously written.

### How to manually use [Test Reconfiguration] feature

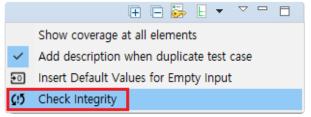
If you click [No] button in [Class Reconfiguration] dialog or [Cancel] button while reconfiguration, following method allows to use [Class Reconfiguration] feature.

• Select [Class Reconfiguration] from the pull-down menu in the [Class Factory View].

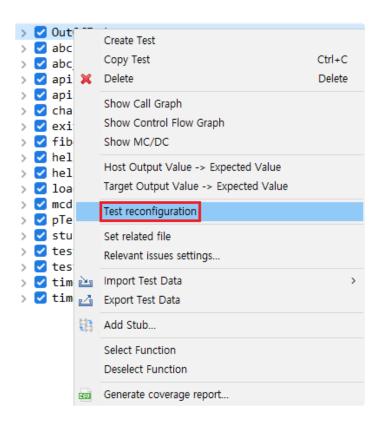


If you click [No] button in [Test Reconfiguration] dialog or [Cancel] button while reconfiguration, following three method allow to use [Test Reconfiguration] feature.

• Select [Integrity Check] in pull-down menu of the Unit Test View.



• Select [Test reconfiguration] to use [Test Reconfiguration] feature in function context menu or test context menu of the Unit Test View.



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

- You can run Reconfiguring Fault Injection from the context menu or from the pull-down menu in the Fault Injection View.
  - Use the Reconfiguring Fault Injection feature in the menu at the top right of the Fault Injection View.



 The Fault Injection View marks fault injection functions that need reconfiguring with a reconfiguration-required status marker . Reconfiguring Fault Injection can be executed by double-clicking or right-clicking on the fault injection function that needs to be reconfigured.

type	e fil	ter t	text			
~	۵	(3)	main()			^
			6:	Remove		Delete
			7:	Reconfig	uring Fault Injection	
			8:	o neconing	annig raan njeedon	_
			9:			
			10:			
				int a= 0;		
				int b = 0	3	
			13:			
		$\checkmark$	<u>14</u> :	while ( b	< 5) {	
		$\checkmark$	<u>15</u> :	a++;		
			16:	}		
			17:			
		$\checkmark$	<u>18</u> :	a =		
			19:	b + 5;		
			20:			
			21:	for (int	i = 0; i < 10; i+	+) 🗸

The fault injection information cannot be modified where the fault injection function with reconfiguration-required status marker.

### 1.2. In Cases of Detected Modification Automatically

When re-analyze or run the tests after modifying source codes, CT 2023.12 detects modifications with integrity checker. The types of source code changes that CT 2023.12 detects are:

- Modifying names of classes used in tests
- Modifying names of test or stub functions.
- Modifying names or type of global variables used in tests.
- · Modifying names of member functions used in class codes
- Modifying name or number of return type or parameter of test functions.
- Modifying the code of the target function to be injected with the fault.

#### Modifying names of classes used in tests

When modifying names of classes used in tests, the Class Reconfiguration dialog shows up.

Class Reconfiguration		– o ×
elect the changed class		
lasses that have been renamed were found. Please select the changed « Classes are sorted according to similarity. Classes with high similarity		3
CommandLineArguments  ➡ CommandLineArguments_mod		
1	Class	File path
· · · · · · · · · · · · · · · · · · ·	CommandLineArguments_mod	C:₩Users₩jslee0202₩Documents₩CT_TEST_src₩ 2 :
	CommandLineTestRunner	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	ComparisonFailure	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	ContainsFailure	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	CompositeTestOutput	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	NormalTestTerminator	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	ConsoleTestOutput	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	MemLeakScopedMutex	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	TeamCityTestOutput	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	🗌 FailFailure	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	SetPointerPlugin	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	CheckFailure	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	JUnitTestOutput	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	MemoryLeakDetector	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	SimpleMutex	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	CrashingTestTerminator	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	CheckEqualFailure	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	EqualsFailure	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	MemoryLeakAllocator	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	□ IgnoredUtestShell	C:\Users\jslee0202\Documents\CT_TEST_src\cpp\cp
	MemoryLeakFailure	C:WI IsoreWisloon202WDocumentsWCT_TEST_srcWcnnWcr

- 1. The left area displays a list of classes where changes have been detected. Functions that have been successfully configured will be marked with  $\checkmark$ .
- 2. The right area displays the list of classes in the current source code:
  - The classes are sorted based on the similarity of their names.
  - Classes with high similarity are automatically linked.
- 3. It allows to search a class name. ( \*: any string, ?: any letter )

#### Modifying names of test or stub functions

When modifying names of test or stub functions, the Function Reconfiguration dialog shows up.

Function reconfiguration		
lect the changed function		
unctions that have been renamed were found. Please select the $\kappa$ Functions are sorted according to similarity. Functions with h		3 4
abc_1()		Ŧ
′ fibonacci_fail() 🕈 fibonacci_failure()	Function name	File path
′ mcdcError() → mcdc_Error()	abc aa()	D:#CT_pri_code#Update_Modify#study_sample.c
′ testMe2()	abc_aa()	D:#CT_prj_code#Update_Modify#study_sample.c
timeOutFunc()	api10	D:#CT_prj_code#Update_Modify#study_sample.c
	apiro	D:#CT_prj_code#Update_Wodify#study_sample.c
	main()	D:#CT_prj_code#Update_Modify#study_sample.c
	pTest()	D:#CT_prj_code#Update_Modify#study_sample.c
	helper()	D:#CT_prj_code#Update_Modify#study_sample.c
	testMe1()	D:#CT_prj_code#Update_Modify#study_sample.c
	timeout()	D:#CT prj_code#Update_Modify#study_sample.c
	loadFile()	D:#CT_prj_code#Update_Modify#study_sample.c
	mcdc Error()	D:#CT_prj_code#Update_Modify#study_sample.c
	stubFunc0()	D:#CT_prj_code#Update_Modify#study_sample.c
	exit error()	D:#CT_prj_code#Update_Modify#study_sample.c
	charlesFunc()	D:#CT_prj_code#Update_Modify#study_sample.c
	helper_Enm2()	D:#CT_prj_code#Update_Modify#study_sample.c
	testMeSecond()	D:#CT_prj_code#Update_Modify#study_sample.c
	time_out_func()	D:#CT_prj_code#Update_Modify#study_sample.c
	fibonacci_failure()	D:#CT_prj_code#Update_Modify#study_sample.c
	OutOfindex_error()	D:#CT_prj_code#Update_Modify#study_sample.c
		D.wei_pj_codewopdate_modifywstady_sample.e

- 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 or type of global variables used in tests

When modifying names or type of global variables used in tests, the Global Variable Reconfiguration dialog shows up.

	0					
ct	Global variable reconfigur	ation				×
Ente	r the changed global v	variable name				
			e name.			
	Missing global variable	global variable name t have been renamed have been found. Please enter the changed global variable name. the deleted global variables.)  Enter current global variable 2  global0				
$\checkmark$	global0 global0					
Ente	r the changed global	/ariable name				
			e name.			
	Missing global variable		Enter current global variable			
	global0		global_var			

- 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.

Enter current global variable
Var_StructA_New[0].a
Var_StructA_New[0].b
Var_StructA_New[0].c
Var_TestA_New
Var_TestB_New

 If many global variables with similar names, such as member variables of a structure, have been changed, you can use the [Change all variables with the same name] feature to modify the global variable names at the same time. If the [Change all variables with the same name] checkbox is checked and the name of a global variable is modified, the name of a global variable with a similar name is modified at the same time. If you uncheck the checkbox, you can edit the names of global variables individually.

#### Modifying names of member functions used in class codes

When modifying names of member functions used in class codes, the Class Code Reconfiguration dialog shows up.

✓ € Turtle_mod (1/1)	<pre>functions, Yellow: Functions changed through reconfiguration, Green: Functions added] 1/// @File: C:\Users\jslee0202\Documents\CT TEST src\cpp\tts-turtle\gmock-turt!</pre>	1/// @File: C:\Users\jslee0202\Documents\CT TEST src\cpp\tts-turtle\gmock-turt
v o a sector (17)  2] Torde,2 (2)	<pre>3 2.tass Concrete_cs_create_gmock_turtle_cpp_Turtle_Turtle_2 : public Turtle_mod 4.public : 5     Concrete_cs_create_gmock_turtle_cpp_Turtle_Turtle_2(){ 6     } 7 8//Function not found. 9/// @Signature: void DeleteMethod() 10/// @Signature:toid DeleteMethod() 11     void DeleteMethod() { 12 13     } 14//Reconfigured function. 15/// @Signature:void Forward_(int distance) 16/// @Signature:void forward_(int distance) 16/// @Signature:void GoTo(int x, int y) 21/// @Signature:void GoTo(int x, int y) 22/// @Signature:void PenDown() 27/// @Signature:void PenDown() 27/// @Signature:void PenDown() 27/// @Signature:void PenDown() 22/// @Signature:void PenDown() 22// @Signature:void PenDown() 22// @Signature:void PenDown() 22// @Signature:void PenDown() 22/// @S</pre>	<pre>/// @file C.C.exter_gmock_turtle_cpp_Turtle_mod_Turtle_2 : public Turtle /// @file C.C.exter_gmock_turtle_cpp_Turtle_mod_Turtle_2(){ //  //  //  //  //  //  //  //  //  //</pre>

- 1. In the left area, the classes that need to be reconfigured are displayed in a tree format.
  - The classes and class codes are structured as a parent-child tree.
  - The number next to each class represents (the number of class codes confirmed by the user / the number of class codes to be reconfigured).
  - The number next to each class code represents the count of changes and deletions in that class code.
- 2. In the right area, you will see the previous class code (gray background, on the left) and the updated new class code (white background, on the right) during the reconfiguration process.
  - The previous class code cannot be modified, while the new class code can be modified.
  - Functions that have been deleted or manually added are indicated with a red background.
  - Functions that have been changed through the reconfiguration process are indicated with a yellow background.
  - Newly added functions are indicated with a green background.

#### 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, the Test Reconfiguration dialog shows up.

Test reconfiguration					– 🗆 ×
Reconfigure test information					
Refer to the previous(left) test information ar $(\!\! \ensuremath{\mathbb{X}}$ Variables with the same name are autom	nd set the new(right) test information. If you drag a variable from the old test and dr atically connected.)	op it on the new test,	the variabl	le is linked and the test data is automatically copied. [Connected: Gree	n, Not Connected: Red]
🗹 🗄 api1(signed int, signed int, f	mcdc_Error(signed int, signed			mcdc_Error(signed int, signed	
<ul> <li>OutOfIndex_error(signed int)_0</li> <li>mcdc_Error(signed int, signed</li> </ul>	Test Info (Update_Modify/mcdcError_test0)		⊿ ≧	Test Info (Update_Modify/mcdc_Error_test0)	<u>تة</u> ك
	Test Structure	Q	e e	Test Structure	Q 🖽 🖻
	Test structure using a tree view and edit the information in the test.			Test structure using a tree view and edit the information in the test	st.
	Name           Test global code           User code           Global Variable           • Test target function           • @ mod.c. growtigned int, signed int)           Local Static Variable           • Decal Variable <th>In Ou</th> <th></th> <th>Name Test global code User code Global Variable</th> <th>In Out A</th>	In Ou		Name Test global code User code Global Variable	In Out A
	Test Info Edit			Test Info Edit	
	Set the number format edit the partition.		^	Manage the stub items added to test.	
	Base © Decimal  O Hexadecimal			Name Description	n Add New Stub Add Stub
1	Variable Partition	Add	· · )		Remove Stub
•	Test Info Test Case Test Code Configuration	66A	· -	Test Info Test Case Test Code Configuration	
L <u></u>					
					Finish Cancel

- 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.

api1(signed int, signed int, f				1	api1(signed int, char, float,		
Test Info (Update_Modify/api1_test0)			<u>a</u> 220		Test Info (Update_Modify/api1_test1)		 2 2a
Test Structure		QE			Test Structure	0	
Test structure using a tree view and edit the information in the test.					Test structure using a tree view and edit the information in the test.		
Name	In	Out	^		Name	In	Out ^
<ul> <li>Test target function</li> </ul>			<ul> <li>Test target function</li> </ul>				
<ul> <li> <sup>T</sup> api1(signed int, signed int, float, unsigned long, long double)      </li> </ul>					<ul> <li>api1(signed int, char, float, unsigned long, long double)</li> </ul>		
Local Static Variable					Local Static Variable		
<ul> <li>Parameter/return</li> </ul>					<ul> <li>Parameter/return</li> </ul>		
○ a : int					o api1_a : int o a : int		
• b : int	~			Т	• b : char		
o c : float	×				<ul> <li>c : float</li> </ul>		
<ul> <li>d : unsigned long</li> </ul>					<ul> <li>d : unsigned long</li> </ul>		
e : long double					<ul> <li>e : long double</li> </ul>		

#### Modifying the code of the target function to be injected with the fault

If the code of the fault injection function has changed, the Reconfiguring Fault Injection dialog appears.

Reconfiguring Fault Injection	
Refer to the old fault injection information(left) and configure the new fault injection information(right). Changed lines are displayed in red. (% Functions with fewer fault injection enabled than before are displayed in orange)	
✓ ● main() type filter text	
✓ 14: while (b < 5) {	
18:         a =           19:         b + 5;	
21: for (int i = 0; i < 10; i++) 21: for (int i = 0; i < 10; i++)	
	~
Inserted Before: Inserted Before:	
1/* test */	~
	~
(Failed) Inserted After. Inserted After:	
1 int testVar = 0;	~
	21
	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
L Name and Andrew Andre	
OK	Cancel

The list of fault injection functions is displayed in area 1, pre-change fault injection information is displayed in area 2, and after-change fault injection information is displayed in area 3.

- The list of fault injection functions
  - If the checkbox is unchecked, the previous fault injection information is retained without saving changes.
- The Fault Injection Information window
  - Pre-change fault injection information can only be copied. You can copy by shortcut(Ctrl + C) or right-click.
  - After-change, the fault injection information can be modified. You can copy/paste by shortcut (Ctrl + C / V) or right-click.
  - Changed lines are marked with a line number in red.
  - Double-clicking on a line selects the same line as the one selected in the other Fault Injection Information window.
  - The code written on the selected line can be shown in the Fault Injection Code window at the bottom.
- The Fault Injection code window
  - The code written before and after the selected line is displayed.
  - Locations where fault injection is not allowed are disabled so that you cannot write code.

If there are no tests generated in the project, or if the fault injection line is enabled but no code is written, the Reconfiguring Fault Injection dialog does not appear.

### 1.3. In Cases of Undetected Modification Automatically

CT 2023.12 cannot detect following types of modification with integrity check.

- Modify value type of global variable that the type is not defined with typedef.
- 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.

### 2. Collaboration Guide

Here's how to share work and results when multiple people collaborate using CT 2023.12.

- Team Testing Usage Guide
- Sharing Projects with Other Users
- Guides to Import Coverages

### 2.1. Team Testing Usage Guide

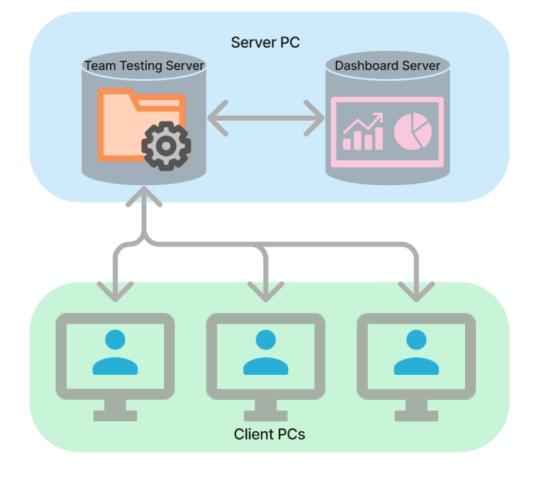
### What is Team Testing?

As time goes by, the scale of software is increasing. With the expansion of software, the scale of testing also grows. Conversely, update cycles are getting shorter, reducing the time available for testing, necessitating more tests in less time.

Team Testing is a method that multiple individuals test a single piece of software. There are several issues with Team Testing. First, it's challenging to share test environments and test cases with others. Second, users create duplicately testing resources like stubs, class codes, etc. Third, merging test results is difficult. Additionally, integrating and rerunning the entire test on one PC to verify the final results adds further complexities.

CT 2023.12 responds to these changes by offering Team Testing capabilities to conduct more tests in less time. Team Testing Server is the server created for CT 2023.12 to provide this feature. During a project, Team Testing Server syncs project configurations among users, enabling sharing of stubs and class codes. Furthermore, it merges test results each time a test is run, displaying them on a dashboard. Users can use the dashboard to monitor project progress at a glance.

In essence, CT 2023.12 is a client where each user conducts tests. Team Testing Server stores and manages projects for sharing among users. The dashboard is a web page presenting project information and progress stored in Team Testing Server.



### Terminology

Below are the terms used in CT 2023.12's team testing:

- Team Project: A project exported to Team Testing Server for collaboration among different users.
- Shared Resources:
  - Configurations that comprise a project, including project properties, preferences, source code, toolchains, as well as resources like stubs and class codes connected to testing.
  - Automatically shared among users.
- Test Resources:
  - Resources necessary for testing, such as tests, test data, etc.
  - Not automatically shared among users; if sharing is needed, they must be imported from Team Testing Server.
- Local: The CT (client) where the user works.
- Team Testing Server: Where the work of all users is merged.
- Commit: The action of reflecting changes made in local onto Team Testing Server.
- Update: The action of importing changes made by other users to the local system.
- Conflict: When updating resources modified and committed by other users, if there are also local changes, it's referred to as a conflict.
- Revision:
  - Modification record in Team Testing Server.
  - Increases when a user commits, changing the version of team project.
  - · Starts at 1 upon project creation and increases by 1 with each commit.

#### **Team Project Process**

The team project can be broadly divided into 3 stages:

- 1. Project Initialization
  - Creating the team project and modifying project configurations to suit the testing environment.
  - Each user imports the team project to their individual PC and collaborates on a shared project.

#### 2. <u>Test Progression</u>

- Tests are carried out through commits and updates.
- Users receive notifications if there are changes in shared resources, allowing them to update.
- If changes in source code make the tests unusable, a re-analysis and test reconfiguration occur.
- Resolving conflicts during the update process is necessary before proceeding with tests.

#### 3. <u>Test Result Merge</u>

- Test results are merged in Team Testing Server and can be monitored in real-time on the dashboard.
- If you need to run the entire test or output the report, import all the tests to one PC and output the report.

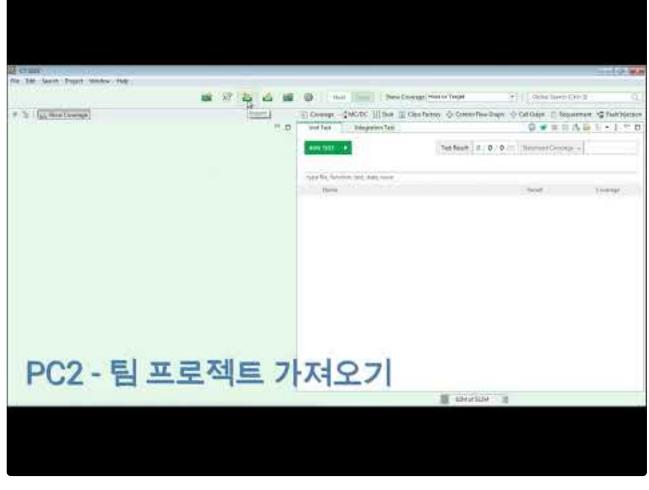
In addition, it also provides <u>On/Offline Mode</u> feature in case the connection to Team Testing Server is unstable.

# 2.1.1. Project Initialization

Using team projects, users can share project environments with others. There are two ways to create a team project. The first is to create a new team project. The second is to convert an existing project into a team project. It's possible to migrate a project used in previous versions to CT 2023.12 and convert it into a team project.

#### **Create and Share a Team Project**

- Connect to Team Testing Server from a single PC to create a team project. Refer to the [Creating <u>a team project</u>]. page in the manual for this process
- 2. Configure the team project to suit the testing environment. Complete toolchain settings and necessary configurations for testing. Then commit the changes to Team Testing Server.
- 3. Import and use the team project on another person's PC.



https://www.youtube.com/embed/Enyc5AFeTho?rel=0

#### **Convert to a Team Project**

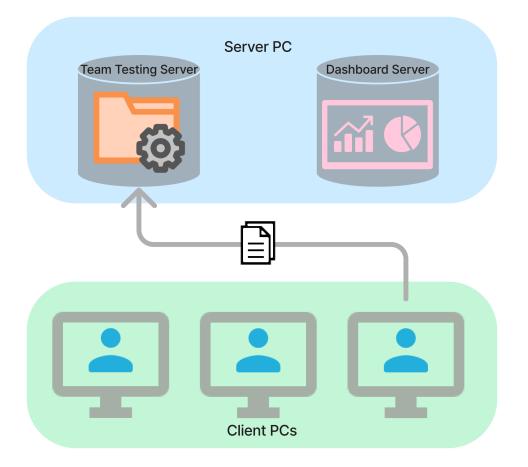
1. Right-click on the project to convert it into a team project.

- 2. If the settings for this project are incomplete, complete the project settings and then commit the changes.
- 3. Import and use the team project on another person's PC.

# 2.1.2. Commit and Update

After multiple users share the team project, testing will be conducted. During the testing process, changes made on the local PC need to be exported to Team Testing Server or changes from Team Testing Server need to be imported into the local PC. Exporting changes from the local PC to Team Testing Server is referred to as commit, and importing changes from Team Testing Server into the local PC is called update.

### Commit

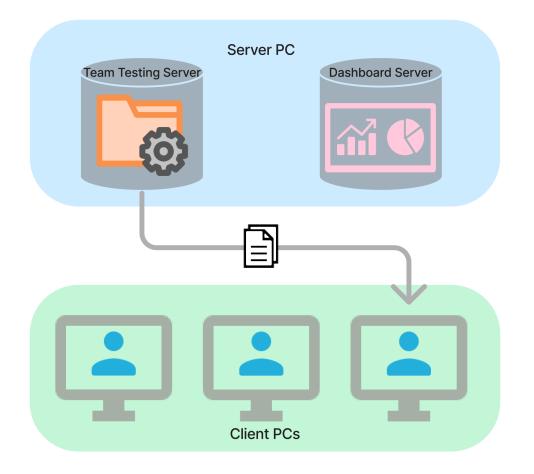


Commit refers to applying the local PC's work to Team Testing Server. The applied work gets shared with other users. Committing with different revisions between Team Testing Server and the local can revert or overwrite other users' changes. When committing, the revisions on Team Testing Server and the local must be the same. If the local's revision is lower than Team Testing Server, an update should be performed before committing.

Users can commit in two ways:

- 1. After running tests, commit changes along with the test results (auto-commit).
- 2. Users can manually commit.
  - When users commit manually, they can verify the contents to be committed in the commit dialog.

### Update



Update refers to import the work from Team Testing Server to the local PC. During an update, encountering resources to be imported from Team Testing Server that have already been modified locally is termed a conflict. When conflicts arise, they are resolved in the update dialog before updating.

The update process is as follows:

- 1. Check and update revisions between Team Testing Server and the local.
- 2. Review resources to be updated in the update dialog. Depending on the resource type, different comparison viewers might be shown the update dialog multiple times.
  - Resources managed in JSON or XML formats, such as project properties or toolchains, can be checked for update content through text comparison.

Update the content of the local (right) reso		testing server (left). For conflicting resources, either directly modify the	local resource and mark it as resolved or copy from the server content.	
Resources	📑 🕒 🔤 🏅	'ext Compare		A 强
Property ctproject juon		Ionteting serve (Non-double)     Tenteting serve (Non-double)     Tenter, "UT_TARL VALUE",     "integerValue": 0     "name": "UT_TARL VALUE",     "integerValue": 5     "integerValue": 5     "integerValue": 1     "integerValue": 3     "integerValue": 0     "integerValue: 0     "integerValu	<pre>// Code</pre>	
		175)	175 )	

• Code-based resources like stubs or class code are checked for update content through source code comparison.

Update			- 0	×
Update Update the content of the local (right) resources to the content from the tea	n testing server (left). For conflicting resources, either directly modify the local resource	e and	d mark it as resolved or copy from the server content.	
Resources 🕞 📴 🗽	C Compare Viewer			_∕\$ %
Tracebolity     Torcebolity     Torceboli	<pre>Commetagewee(Non-eduble) 1/// Bindsubset bind Auto Generated Input Macro 1/// Bindsubset ULOng adlers1_out 2/// Bindsubset ULOng adlers1_out 2/// File: C:\Usera\Choo Seywon\AppBata\Rosming\CodeSerc 2 /// File: C:\Usera\Choo Seywon\AppBata\Rosming\CodeSerc 2 </pre>		<pre>/ tool /// Bigmature: uLong adler32_combine_tuLong adle /// Bigmature: uLong adler32_combine_tuLong adle /// Bile: GiUDera\Choo Bayoon\AppData\Roaming\Co /// Eile: GiUDera\Choo Bayoon\AppData\Roaming\Co // Eile: GiUDera\Choo Bayoon\AppData\Roaming\Co /// Eile: GiUDera\Choo Bayoon\AppData\Roaming\Co /// Eile: GiUDera\Choo Bayoon\AppData\Roaming\Co // Eile: GiUDera\Roaming\Co // Eile: GiUDera\Roami</pre>	ero A
Added: 0 ₩ Deleted: 0 ≠ Modified: 0 ↔ Conflicts: 3				
			ОК Са	ncel

• The resources managed by the DB, such as requirement traceability, compare the data in the DB in text format to check for updates.

pdate Jodate the content of the local (right) resources to the content from the	team testing server (left). For conflicting resources, either directly modify the local resource	and	mark it as resolved or copy from the server content.	
esources 📑 险 🔤	Text Compare			₰
Tracebility     Tracebility     Tracebility     Sub-     Tracebility     Sub-     Tracebility     Sub-     Sub-	<pre>If you want the start of this area or modify it out 1 if you modify the content of this area or modify it out 3 REO REY, IEST, IESTCASE NUME, IEST IN, IEST IVE 4 International Content of the start of the start and an area of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start ADS, start of the start of the start of the start of the start ADS, start of the start of the start of the start of the start ADS, start of the start ADS, start of the start</pre>	)-e-  }>`	<pre>/ Load / If you modify the content of this area or modify it is the traceability information file will be corrupted as 0 mer. rest, restrate with, rest 10, rest // restrate and rest is a second second second // restrate and rest is a second second second second // restrate and rest is a second second second second // restrate and restrate and restrate and rest // restrate and restrate and restrate and restrate // restrate and restrate and restrate and restrate and restrate // restrate and restrate and restrate and restrate and restrate // restrate and restrate // restrate and restrate and</pre>	
Added: 0 # Deleted: 0 / Modified: 0 + Conflicts: 3				

• For fault injections, the update content is reviewed in a form similar to Fault Injection view.

Update		- 0 X
Update		
	am testing server (left). For conflicting resources, either directly modify the local resource	and mark it as resolved or copy from the server content.
	😭 Team testing server (Non-editable)	🖉 Local
		type filter text v ⊕ (1) add#32(unsigned long, const unsigned char', unsigned int) v ⊕ (1) add#32(unsigned long, const unsigned char', unsigned int) v ⊕ (1) add#32(unsigned long, const unsigned char', unsigned int) inserted Before: 1 inserted Before: 1 inserted After:
		norred Ater:
♦ Added: 0 ¥ Deleted: 0  Modified: 0   Conflicts: 3		4
		OK Cancel

• For tests, the update content is reviewed in a test editor format.

ests	📑 🕒 🔤 🍒	Team testing server (Non-editable)			🖉 Local		
E Unit Test     adler32 combine64 test1	adler32_combine_(unsigned long		adler32_combine_(unsigned long				
adler32_combine_test1		Test Info (zlib/adler32_combinetest2)		총 🏼 🖂	Test Info (zlib/adler32_combine_test1)	ł	t 🛆 i
		Test Structure		Q E E	Test Structure	C	
		Test structure using a tree view and edit the information	in the test.		Test structure using a tree view and edit the information in the test.		
	Name Teme global code User code Global Varible > Test target function User code > Stub Test Info Edit	In 	Out	Name Test global code User code Global Vauble S Test target function User code S Stud Test time failt Test Info failt	In	Out	
Addect: 0, M. Deletect: 0, # Modified: 0, ++ 0	anflicter 2	Test Info Test Case   Test Code   Configuration			Text Info Text Case Text Code Configuration		

- 3. Resolving conflicts can be done in two ways:
  - · Save as server version
    - Select the local from server to local) icon to copy the version from Team Testing Server to the local. Copying from Team Testing Server to the local overwrites local changes with Team Testing Server version, and local changes cannot be modified. The like (cancel copy) icon can revert the copied version back to its original state.
  - Manually resolve by editing
    - Without copying the version from Team Testing Server, the user directly edits the local version. Referring to the current local and Team Testing Server versions, the user saves the version they've edited. When users resolve conflicts manually, they should select the (mark as resolved) icon to indicate the resolution of the conflict for that resource.
- 4. When all conflicts are resolved, the [OK] button becomes active. Clicking [OK] proceeds with the update.
- 5. Depending on the type of updated resources, reanalysis might be necessary after the update.

During team testing, update notifications are provided to keep the local version up-to-date. Besides update alerts, users are informed of the necessity for an update if Team Testing Server and the local have different revisions when committing.

### Modify shared resources

Modifying shared resources like stubs or class codes can alter the results of connected tests. When committing the changed resources, the results of the test in Team Testing Server will be unreliable, and the results of the test connected with those resources will be deleted from Team Testing Server. In such cases, the project's coverage might decrease.

```
void func() {
    if( returnNum() == 1 ) {
        /* ... */
    } else {
        /* ... */
      }
}
```

Creating a stub for returnNum() in the above code and connecting it to a test for void func() can affect coverage based on the return value of returnNum() stub. When modifying shared resources connected to tests, the linked tests should be rerun. Committing modified resources leads to the deletion of execution results of tests connected with those resources from Team Testing Server. Tests with deleted results are managed as 'need-to-rerun test'. During an update, if there are need-to-rerun tests, they are marked with ! in the [Unit/Integration Test] view. Rerunning need-to-rerun will commit new results and remove the ! marker.

When modifying shared resources and automatically committing after running the connected tests, the executed tests are not managed as need-to-rerun tests since new test results have been committed.

### **Reference Videos**

Here are videos you can refer to regarding commit/update/conflict:

• A video demonstrating the creation of stubs, executing tests for automatic committing, and updating committed stubs from another PC.

air deno	10 17 Ja d	and the second the transport	wick whole will a	244 June 199-5	1
<ul> <li>V. Scherowski</li> </ul>		() (mmp: -(14000) (21100 (2100 h)			A CONTRACTOR OF A CONTRACTOR
	- 0	Unit fiel ( ) integration ( less )	NUMBER OF BUILDING	THE REAL	315
		Environmente	10000001010000001000	10110-000-01	
		Type too A name in the paper south			
		The			transp.
		- 2 4018	070		
		1 W. WEILD States and Discontinues	11/18		
		1 P will wall started best man of the	1.5 10		
		and and many star. Assai	10.04	- # · ·	
		) 著 igithy Gr L, Bail A. 'g	10.0		
		1 # 2014 (1000 - 1000 %, 1000 + 1000 )			
		a 20 month that "I thank they be a set	110		
		a W apd7 ( std., () second the st	- 10 A		
	and the second se	to be applied to the second states while		- W. H.	
THE DESIGN OF THE OWNER.	N 44 4 - C 1 - C	a R. and Thomas A. Suby Streams, 1994			
Asnage shitl dents of literary proget, 21 literary	the second s	1 2 bool test and field runn half			
Same while must	+ This cole - Carilyauther	A REAL PROPERTY AND A DESCRIPTION OF A DESCRIPTION OF	10.7 B		
- + printinghed in the sensitive upnet very	1.11) Belletr dage Riderther	A Part of the second se	0.78		
1. CARLEN AND A CONTRACT OF A CONTRACT OF	Z()/ BilgerCarter unter second Det 3, that	a water and a line of the second second	(iii) iii		
	4.1x7 #Highertertailer hiddelerter 4.1x7 #Highert Forenegerthemationeg Hadde	A Realized and a second sec	10.10		
	All war attain the	+ M mind British ( 1991)		18.0	
	Scelars &				
	1983 Barris Martinessen				
4	1107 - Searce bey personal takent station beyound its				
	end of the second distribution of the second s		PROVIDENCE IN		

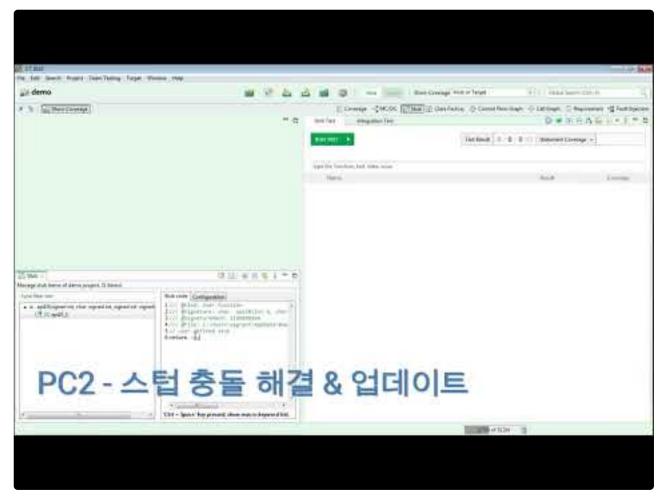
https://www.youtube.com/embed/77Dpg8CYoeo?rel=0

• This video showcases manual commit after modifying source code and updates via update notifications on another PC.

載 towner		0 N	P.
Connett	fails will the load memory single and comber are difference. (Deleng the "Dd" bottor will posts the loss		1
	has no on one would that the statement of the statement o		1
will shale		A 14	1.
Cont	<pre>#lectudeint(0, h) #lectudeint(0, h) #lectudei</pre>	<pre>     # decision create in     # decision</pre>	
	<pre>inade( int eq:the* wig() ) int.L = T (     prior()'equation int pairs int equthe* eq:() )) int     prior()'equation int pairs int equthe* eq:() )) int     prior()'equation int ()     if ()</pre>	<pre>1 1 1 1 1 1 1 1 1 1 1 1 1 1</pre>	
6 Most 8 + Deniel 8 4 Matrix: 1		Annumation to previous	
		DK [ C204	
a ere		ME ANALYSIS (B)	

https://www.youtube.com/embed/k6801k-71BQ?rel=0

• A video demonstrating the process of modifying user stubs, committing changes, and resolving conflicts arising during updates.



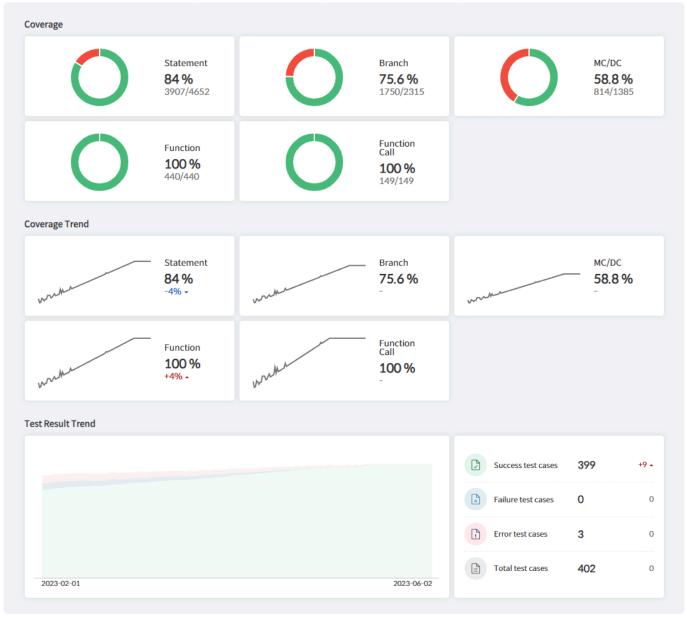
https://www.youtube.com/embed/67QPhrysG5k?rel=0

# 2.1.3. Test Result Merge

After all tests for the team project are completed, it's necessary to merge the results. There are two ways to check the aggregated results. The first is to check them on the dashboard. When running tests for the team project, Team Testing Server merges the results of all tests and displays them on the dashboard. The second is to generate reports for checking. Currently, CT 2023.12 does not provide report generation from Team Testing Server, so all tests must be gathered on one PC to generate a report.

### Dashboard

You can view the results merged on Team Testing Server in a summary from the dashboard.



#### **Report Generation**

You can inport all tests onto a single PC and generate reports for those tests.

1. Use [Import Team Test] feature to import tests from Team Testing Server.

- 2. Since importing team tests doesn't bring test results, all tests need to be run again.
- 3. Use [Generate test report] feature to create a report for the project.

# 2.1.4. On/Offline Mode

When testing in an environment where Team Testing Server is offline or the network connectivity is unstable, it can be inconvenient due to the connection retry dialog. In such cases, you can switch the project to offline mode for usage.

Progress Information	
Connecting to Team Testing server	
Retrying connection (3/3)	
	Cancel

To prevent repeated dialog for connection retry in a unstable network connection, you can disconnect from Team Testing Server and operate in offline mode. Toggle between online and offline modes in [Properties] > [Team Testing] > [On/Offline].

Properties for sadfasdf		— 🗆 X	
type filter text	On/Offline		
Inclusion and Exclusion Info Source File Types Target Test Team Testing Delete Project On/Offline > Test	© online () offline	Restore Defaults Apply	
	D	Apply and Close Cancel	]

In offline mode, you cannot perform tasks that require connection with Team Testing Server, such as

committing or updating. In essence, running tests in offline mode does not commit the results. Once the connection with Team Testing Server become stable, you can switch back to online mode. When switch to online mode, resource names are synchronized to prevent overlaps with server resources.

Additionally, an update is carried out to match the latest revision from Team Testing Server. To ensure your local resources stay up-to-date, always perform an update before continuing your work.

As offline mode doesn't synchronize with Team Testing Server, it's advisable to use it for the shortest time possible.

## **2.2. Sharing Projects with Other Users**

You can share the CT 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

# 2.2.1. (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

## 2.2.1.1. Export project

You can export projects, including project setup and testing.

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

File	Edit	Search	Project	Target	Window
	New				>
	Close			C	Ctrl+W
	Close	All		Ctrl+Sł	nift+W
	Save				Ctrl+S
Ū.	Save A	All		Ctrl+S	hift+S
	Renan	ne			F2
	Refres	h			F5
	Switch	n Worksp	ace		
2	Impor	t			
4	Export	t			
	Prope	rties		Alt	+Enter
	Exit				

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

Export	
Select Export the project, including project settings and tests.	ß
Select an export wizard: type filter text	
<ul> <li>Coverage</li> <li>General</li> <li>Export Project</li> <li>Metrics</li> <li>Preferences</li> <li>Test</li> </ul>	
< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel

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

I Export Project	—		×
Export Project			
Export the project, including project settings and tests.			
Project:			
☑ ≌ study_sample		Select Deselec	
1 of 1 selected. Options Include Toolchain			
Path: C:#Users#hikim#Desktop		Sear	ch
< Back Next > Finish		Cance	I

- (Ver.3.7 or later) When you export a project, you can export it including the toolchain and source files.
- 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.

## 2.2.1.2. 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.

File	Edit Se	arch	Project	Target	Window
	New				>
	Close			C	Ctrl+W
	Close All			Ctrl+Sh	nift+W
	Save			(	Ctrl+S
G	Save All			Ctrl+S	hift+S
	Rename				F2
	Refresh				F5
	Switch W	orkspa	ace		
2	Import				
4	Export				
	Properties			Alt	+Enter
	Exit				

- 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.

Import						×		
Import a Project Import the export	ed project.							
Project directory: C:#Users#hikim#Downloads#test_vs2013_tch_src_export_imported2_20220428102501								
Project name:	ne: test_vs2013_tch_src_export_imported							
Location:	C:\Users\hikim\Desktop\ct_work	kspace						
Select Toolchain								
Default	Toolchain		Description					
	hyeintoolchain targetToolchain GCC 4.7 (32bit) GCC 5.3 (32bit)		자동으로 생성되었습니 자동으로 생성되었습니					
Options ☑ Include Toolch: ☑ Include Source					Toolchain	Setting		
		< Back	Next >	Finish	Cance	el		

• (After Ver.3.7) If the directory contains a toolchain or source codes, its options are checked automatically.

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 User Manual.

- 5. Click the [Next] button.
- 6. 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.

Import	
Check the paths included in the project	
🚯 Invalid path: 15/15	
Path(click to edit):	-
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\adler32.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\crc32.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\deflate.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\gzclose.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\gzlib.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\gzread.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\gzwrite.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\infback.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\inffast.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\inflate.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\inftrees.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\trees.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\uncompr.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.11\zutil.c	
C:\Users\seo hyunji\Desktop\target plugin\zlib-1.2.11\zlib-1.2.13\compress.c	
< Back Next > Finish	Cancel

7. 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.

Import	- • ×
Check the paths included in the project	
() Invalid path: 0/15	
15 paths have been resolved.	
Path(click to edit):	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\adler32.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\compress.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\crc32.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\deflate.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\gzclose.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\gzlib.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\gzread.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\gzwrite.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\infback.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\inffast.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\inflate.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\inftrees.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\trees.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\uncompr.c	
C:\Users\vagrant\Desktop\source\zlib1211\zlib-1.2.11\zutil.c	
< Back Next > Finish	Cancel

\* If is not in absolute path Windows format, the path is not checked for validity.

8. 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 Env	vironment Settings					
		te the source code with the test code. You can run the target test required fields on the Build/Run tab.				
GNU Co	ompilers > gcc > 5.3 > others	nodebugger <u>Import Target Environment Settings</u>				
Settings						
Analysis	Name	Value				
Build	language	c				
Run	Toolchain Name	GCC 5.3 (32bit)				
etc.	Status	This toolchain is supported.				
ctc.	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:					
	b comparent a					
		< 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.

## 2.2.2. (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
- <u>RTV server user guide</u>

## 2.2.2.1. 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 CT 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 CT workspace path that he uses.
- Select top-level path to the project directory to import the projects using [Import] > [General] > [Existing Projects into Workspace] function.

Import -	
Import Projects Select a directory to search for existing Eclipse projects.	
Select root directory: F:\CT3.2\6-2	Browse
O Select archive file:	Browse
Projects:	
phm_RTVPROJ (F:\CT3.2\6-2\phm_RTVPROJ)	Select All
phm_rtv (F:\CT3.2\6-2\phm_rtv)	Deselect All
	Refresh
Copy projects into workspace	
< Back Next > Finish	Cancel

- 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 CT 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.

```
Suresofttech
```

### 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 CT workspace.
- 2. The user who wants to share the project connects to the same RTV server where the above project was created from CT 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 CT 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
- plugins/org.eclipse.equinox.launcher\_1.4.0.v20161219-1356.jar --launcher.library 2
- 3
- 4 plugins/org.eclipse.equinox.launcher.win32.win32.x86 64 1.1.500.v20170531-1133 5 -data
- 6 @noDefault -q
- C:\temp 8
- 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.

ort tests				
xport rests xport project's test informations.				
,				
export path: \$(temp_folder)		Searc	h	
Unit Test				
✓ Test				
☑ Test Data				
Integration Test				
✓ Test				
🗹 Test Data				
Stub				
O Connected Stub				
● All Stub				
Option				
Overwrite existing test files without warning				
Export only checked tests in Unit/Integration Test View.				
< Back Next > Finish	1	Cancel		
			_	
Import tests	_		×	
	-		×	
Import tests port tests mport project's test informations.	-		×	
aport tests	-		×	
mport tests mport project's test informations.	-			
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i>	-	Searc		
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test	-			
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test	-			
pport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data	-			
pport tests mport project's test informations.  mport path: <i>\$(temp_folder)</i> Unit Test ☐ Test ☐ Test Data Integration Test				
pport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test				
pport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test ☑ Test Data				
pport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test ☑ Test Data Stub				
port tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test Data Stub ◯ Connected Stub				
pport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test ☑ Test Data Stub				
port tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test Data Stub ◯ Connected Stub				
port tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test Data Stub ○ Connected Stub ④ All Stub				
port tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test Data Stub ○ Connected Stub ◎ All Stub Option				
pport tests mport project's test informations. mport path: \$(temp_folder) Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test Data Integration Test ☑ Test Data Stub ○ Connected Stub ④ All Stub Option □ Overwrite existing test files without warning				
apport tests         mport project's test informations.         mport path:       \$(temp_folder)         Unit Test         Test         Test Data         Integration 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.				
apport tests         mport project's test informations.         mport path:       \$(temp_folder)         Unit Test         Test         Test Data         Integration 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.				
apport tests         mport project's test informations.         mport path:       \$(temp_folder)         Unit Test         Test         Test Data         Integration 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.				
apport tests         mport project's test informations.         mport path:       \$(temp_folder)         Unit Test         Test         Test Data         Integration 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.			h	

## 2.2.2.2. 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).

## 2.3. Guides to Import Coverages

When importing coverages from CT 2023.12 in another environment or COVER, these are imported by the following three criteria. If the criteria are not met, coverage imports may fail.

- version of coverage shared file
- ternary operation option
- <u>coverage type</u>

## 2.3.1. Import Coverages by Version

When CT 2023.12 import coverages, it checks the version of the coverage shared file.

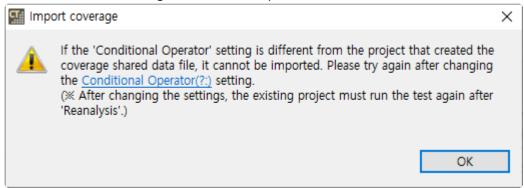
• When importing the higher-versioned coverage shared file, the coverage cannot be imported.

CT Impo	ort coverage	$\times$
	It cannot be imported because it is a higher version coverage shared data file. Please check the version of the tool that exported the coverage shared data file.	
	ОК	

• When importing the lower-versioned coverage shared file, importing the coverages for some functions may fail depending on the option of the tool that exported coverages.

# 2.3.2. Import Coverages by Conditional Operation Option

When the conditional operation option of CT 2023.12 differs from the tool that exported the coverage shared files, the coverages cannot be imported.



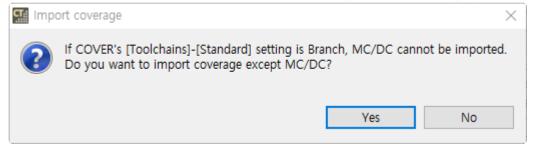
Click the link in the warning window or [Preferences] > [Test] > [Coverage] > Branch coverage, MC/DC measurement operator]. Then, change the [Conditional Operator(?:)] option to match the file you want to import.

- When [Toolchains] > [Standard] in COVER is [COVER] > [Branch], turn off the [Conditional Operator(?:)] option.
- When [Toolchains] > [Standard] in COVER is [COVER] > [MC/DC], turn on the [Conditional Operator(?:)] option.

When changing the option, run the tests again after [Reanalyze].

## 2.3.3. Import Coverages by Coverage Type

After Controller Tester 3.6, users can import coverages when coverage types are different.



When selecting [Yes], statement/branch coverages are imported except MC/DC. When selecting [No], coverages are not imported.

## 3. Scenario(Time-based) Test Usage Guide

During the requirement testing, you may come across the following requirement.

When the door of the car, which was open, closes, the interior light stays on for 5 seconds and then turns off.

When there is a timer function for this, this timer function should be repeatedly called at set intervals to check whether the interior light turns off after 5 seconds.

There are situations where you need to test functions that are called periodically. After CT 2023.12, a scenario testing feature has been introduced to support these types of tests.

#### **Conditions for Scenario Testing**

To convert a normal test into a scenario test, all of the following conditions must be met.

- C Project
- The test target functions must only have functions with void return type and no parameters

#### **Convert to Scenario Test**

In [Test Editor] > [Test Info Tab] > [Test target function], select [Convert to Scenario test] to convert to a scenario test.

E TASK_1ms()_0 ×					- 0
Test Info (scenario/TASK_1ms_test0)					🌲 👍 🖄
Test Structure Test structure using a tree view and edit the information in	the test			o add, delete and change the order	
Name	In	Out	Function	Path	Add
Test global code User code			TASK_1ms()	D:\CT_prj_code\periodic\ta	
Global Variable			-		Remove
> Test target function					Up
User code					Down
> Stub			-		
			-		Set related file
			Convert to Scenario t	est	
Test Info Test Case Test Code Configuration					

A Once converted to a scenario test, it cannot be reverted back to a normal test.

Scenario tests maintain the test case context. That is, it runs the next test case while maintaining the state of the previous test case.

#### **Scenario Test**

JJC I	nfo (scenario/TASK_1ms_test0)				i di 1
	Structure tructure using a tree view and edit the info	C ormatio	、 ⊞ 🖃 n in the	Test Info Edit Test target functions to add, delete and	change the order
Nam	ne Test global code	In	Out	Cycle unit settings: ms	
	User code Global Variable			<ul> <li>Initial function</li> <li>initial(signed int, signed int,</li> </ul>	Add initial function
~	Scenario Variable • Duration : unsigned long long	<b>~</b>		<ul> <li>Cycle function</li> </ul>	Add cycle function
~	<ul> <li>Total Duration : unsigned long long Test target function</li> </ul>			✓ ● <sup>S</sup> TASK_1ms() Start time: 0 ms	Remove
>	<ul> <li>initial(signed int, signed int, unsignec</li> <li>TASK_1ms()</li> </ul>			Cycle time: 10 ms	Up
,	User code				Down
>	Stub				Set related file

Two variables are added to assist with scenario testing.

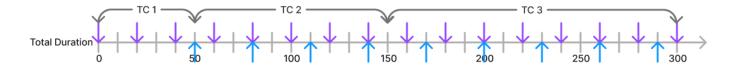
- Duration
  - You can enter values in [Test Case Tab].
  - $\circ~$  It determines the number of iterations of the for loop for each test case.
- Total Duration
  - It shows the cumulative number of repetitions of the for loop in the test case.

Among the test target fucntions, two settings are added to the cycle function:

- Start time
  - This is the time at which the function calls start.
  - If the start time is 40, the function won't be called when the total duration is between 0 and 39.
- Cycle time
  - This is the time interval for repeating function calls.
  - If the period time is 10, the function will be called when the total duration is 10, 20, 30, and so on.

Below is an illustration to help understanding the scenario testing concept.

			тс	Duration	Total Duration
	start time	cycle time	1	50	50
20ms_Task	0	20	2	100	150
30ms_Task	50	30	3	150	300



The cycle unit can be set in ms,  $\mu$ s, or ns. This value is only displayed in the test report and test editor, not affecting the actual execution time. Therefore, even if the cycle unit is set to ms and the total duration is 5000, it won't be executed for an actual 5 seconds. Scenario testing simulates this time for running tests.

#### **Test Code of Scenario Test**

Based on the settings in [Test Info Tab], the test code is generated as follows: The initial function is a function that is called only once before iterating through a for loop, and it's initially invoked in the first test case. After that, it isn't called in any other test cases.

```
/* Declaration (parameter/return/target object) variables */
unsigned int CS TC SPENT TIME = 0; // Stores the Duration input value.
static unsigned int CS TOTAL SPENT TIME = 0; // Stores the number of repetitio
ns of test cases. The value of this variable is stored as the output of Total
Duration.
/* Input */
CS TC SPENT TIME = CS INT INPUT (unsigned int, "CS TC SPENT TIME");
/* Call initial function */
if (CS TOTAL SPENT TIME == 0) {
    // Position where initial function is added
}
// Where cycle functions are repeated. Loops for the Duration input value.
for (int CS CYCLE INDEX = 0; CS CYCLE INDEX < CS TC SPENT TIME; CS CYCLE INDE
X++) {
    if ((CS TOTAL SPENT TIME >= 0) && ((CS TOTAL SPENT TIME - 0) % 10 == 0)) {
        /* TASK 1ms() */
        TASK 1ms();
```

```
}
CS_TOTAL_SPENT_TIME++;
}
/* Output */
CS_INT_OUTPUT(CS_TOTAL_SPENT_TIME, "CS_TOTAL_SPENT_TIME");
```

#### **Examples**

You can fulfill various requirements by using scenario testing.

- Check for changes in specific variables during a scenario test run
- Determine whether to call a function based on the value of the global variable

# 3.1. Check for changes in specific variables during a scenario test run

Divide the test cases to check when running a scenario test to verify the value of a specific variable during the test run. For instance, in a test where the cycle unit is ms, if we want to check the changes in a specific variable at 2 and 4 seconds, we design the test cases as follows:

Test Case (TC)	Duration	Total Duration
1	1999	1999
2	1	2000
3	1999	3999
4	1	4000

Check the value change before/after 2 seconds through TC3 and TC4 and check the value change before/after 4 seconds through TC3 and TC4.

By dividing the test cases in this way, we can check desired values at specific time. We will explain it in detail with a simple example.

### **Source Code and Requirement**

The source code and requirement to be used as an example are as follows.

```
#include <stdio.h>
#include <stdbool.h>
typedef enum {
    CLOSED, OPEN
} OpenCloseState;
typedef enum {
    OFF, ON
} OnOffState;
OpenCloseState doorState;
OpenCloseState doorSensor;
OnOffState ignitionState;
OnOffState lightState;
void initial() {
        doorState = OPEN;
        doorSensor = OPEN;
    }
}
```

```
ignitionState = OFF;
        lightState = OFF;
}
void lightOn() { if (lightState != ON) lightState = ON; }
void lightOff() { if (lightState != OFF) lightState = OFF; }
void setDoorSensor(OpenCloseState sensor) {
        doorSensor = sensor;
}
void tick() {
   static int timer = 0;
   if (doorState == OPEN && doorSensor == CLOSED) {
        timer = 500;
       lightOn();
    } else if (ignitionState == ON) {
       timer = 0;
        lightOff();
    }
   if (timer > 0) timer--;
   if (timer == 0) lightOff();
    doorState = doorSensor;
}
```

Requirement: When the door of the car, which was open, closes, the interior li ght stays on for 5 seconds and then turns off.

#### **Test Design**

We will design a test that meets these requirements.

h3 Design of Cycle Function and Cycle Unit

Using The indoor light turns off after being on for 5 seconds and timer = 500;, it can be understood that the time unit of the timer is ms, and the function "@tick()@" is called every 10ms. Therefore, set "@tick()@" as the cycle function, the cycle unit as ms, the start time of "@tick()@" as 0, and the cycle time as 10.

h3 Design of Initial Function

The initial() function initializes each sensor and state. Both doorState and doorSensor are initially set to OPEN. To satisfy the requirement When the door of the car, which was open, c loses, set the parameter value in the setDoorSensor() function to CLOSED.

h3 Design of Test Cases and Scenario Variable (Time)

To verify the requirement, check the output value of lightState. The interior light should remain on for 5 seconds and then turn off. This means that lightState remains ON until the 499th call of tick(), and lightState changes to OFF on the 500th call. To check this, set the time to 4990 for the first test case and 10 for the second test case. Select lightState in the global variable, check [Output], run the test, and verify the value of lightState.

#### Write Test

Based on the above design, writing tests will look like the following:

• [Test Info Tab]

Name In Out	-
<ul> <li>• lightState : OnOffState</li> <li>• lightState : OnOffState</li> <li>&gt; Scenario Variable</li> <li>• Test target function</li> <li>&gt; @ initial()</li> <li>• @ setDoorSensor(OpenCloseState)</li> <li>Local Static Variable</li> <li>• Parameter/return</li> <li>• sensor : OpenCloseState</li> <li>• sensor : OpenCloseState</li> </ul>	al function nitial() etDoorSensor(OpenCloseState) le function
Before call code After call code Auto-generated Variable Solution User code Stub	

• [Test Case Tab]

Parameter	Туре	Input	Expected Value	Host Output	Target Outp
<ul> <li>lightState</li> </ul>	OnOffState		ON(1)		
<ul> <li>sensor</li> </ul>	OpenCloseState	CLOSED(0)			
<ul> <li>Duration</li> </ul>	unsigned int	4990			
<ul> <li>Total Duration</li> </ul>	unsigned int				
est Case (scenario/tick_test0) #	2			Q	16 🗉 🔶 d
est Case (scenario/tick_test0) #		Input	Expected Value	1	₩ 📰 🔶 😅
	2 Type OnOffState	Input	Expected Value OFF(0)	् Host Output	
Parameter	Туре	Input		1	
• lightState	Type OnOffState	Input 10		1	표 🔝 🔶 ၎ Target Outp

#### **Checking Test Results**

Run the test and verify the results in [Test Case Tab].

Test Case (scenario/tick_test0) #	t <b>1</b>			Q	16 1 <b>6</b> 4
Parameter	Туре	Input	Expected Value	Host Output	Target Outp
<ul> <li>lightState</li> </ul>	OnOffState		ON(1)	1	
<ul> <li>sensor</li> </ul>	OpenCloseState	CLOSED(0)			
<ul> <li>Duration</li> </ul>	unsigned int	4990			
<ul> <li>Total Duration</li> </ul>	unsigned int			4990	
Fest Case (scenario/tick_test0) #		1t	Europhal Malue		
Parameter	Туре	Input	Expected Value	Nost Output	Target Outp
<ul> <li>lightState</li> </ul>	OnOffState		OFF(0)	0	
<ul> <li>sensor</li> </ul>	OpenCloseState				
<ul> <li>Duration</li> </ul>	unsigned int	10			

During 0ms to 4990ms, the lightState is ON(1), and when it reaches 5000ms, the lightState changes to OFF(0). By using scenario testing, it has been confirmed that the source code meets the requirements.

# 3.2. Determine whether to call a function based on the value of the global variable

When the value of the global variable determines whether a function is called, it is tested using [Before/ Atfer call code].

```
// Before call code
if( globalVar == 1 ) {
// Function call code.
func()
// After call code
}
```

In this way, add an if statement in the code before and after the function call to invoke the function when specific conditions are met. I'll explain in detail using examples of traffic lights and sound signals.

#### **Source Code and Requirements**

The source code and requirements to be used as an example are as follows.

```
#include <stdio.h>
typedef enum {
   RED,
    GREEN
} TrafficLightState;
typedef enum {
        ON,
        OFF
} SoundSystemState;
TrafficLightState trafficLight;
SoundSystemState soundSystem;
void init() {
        trafficLight = GREEN;
        soundSystem = OFF;
}
void setSoundSystem (SoundSystemState state) {
        soundSystem = state;
}
```

```
void alarmForBlind() {
        if ( trafficLight == GREEN ) {
                printf("beep for blind\n");
        } else if( trafficLight == RED ) {
                printf("warning for blind\n");
        }
}
void tick() {
   static int timer = 50;
        if(timer > 0) {
                timer--;
        }
        if( timer == 0 ) {
                timer = 50;
                if ( trafficLight == GREEN ) {
                        trafficLight = RED;
                } else if( trafficLight == RED ) {
                        trafficLight = GREEN;
                }
        }
}
Requirement: The red and green lights are each on for 5 minutes.
If the sound signal is on and the traffic light is green, a signal for visuall
y impaired individuals is output once per second.
If the sound signal is on and the traffic light is red, a warning for visuall
y impaired individuals is output once per second.
```

#### **Test Design**

We will design a test that meets these requirements.

h3 Design of Cycle Function and Cycle Unit

In the tick() function, the timer of the traffic light is set to 50, and as per the requirement, each light stays on for 5 seconds. Therefore, the cycle unit is ms, and the cycle time for tick() is 100ms. Since the sound signal outputs a signal every 1 second, the cycle time for alarmForBlind() is 1000ms (1 second).

h3 Design of Initial Function

Call the initial () function as an initial function to assign initial values to each sensor and state. At this point, the initial value of trafficLight is GREEN and the initial value of soundSystem is OFF.

#### h3 Design of Test Cases and Variables

To verify the requirements, we'll turn on the sound signal and check signals when the light is blue or red. Accordingly, we'll design the test cases as follows:

тс	Time	Sound Signal
1	5000	OFF
2	5000	OFF
3	5000	ON
4	5000	ON
5	5000	OFF
6	5000	OFF

To change the sound signal value, input the following into the user code. When the test case is 3, turn on the sound signal, and when the test case is 5, turn it off.

```
if (CS_TESTCASENO()==3) {
    setSoundSystem(ON);
} else if (CS_TESTCASENO()==5) {
    setSoundSystem(OFF);
}
```

Select trafficLight from the global variables and check [Output] to confirm the status of the traffic light.

#### Write Test

Based on the above design, writing tests will look like the following:

• [Test Info Tab] > [User code]

tick()_0 ×			
Test Info (crosswalk/tick_test0)			총 凸 관
Test Structure		Q ⊞ E	Test Info Edit
Test structure using a tree view and edit the infe	ormatio	n in the test.	Edit code to be executed at the beginning of the test function
Name	In	Out	1 if (CS TESTCASENO()==3) {
Test global code			<pre>2 setSoundSystem(ON);</pre>
User code			<pre>3 } else if (CS_TESTCASENO()==5) {</pre>
<ul> <li>Global Variable</li> </ul>			4 setSoundSystem(OFF);
✓ ● trafficLight : TrafficLightState			5 }
trafficLight : TrafficLightState			

• [Test Info Tab] > [Test target function]

L: tick()_0 ×			- 8
Test Info (crosswalk/tick_test0)			🌲 🕹 èi
Test Structure Test structure using a tree view and edit the infor Name Test global code	the test.	Test Info Edit This is an editor for scenario test. Settings	
User code  Global Variable  v o trafficLight : TrafficLightState		Cycle unit settings: ms	Add initial function
trafficLight : TrafficLightState     Scenario Variable		<ul> <li>init()</li> <li>✓ Cycle function</li> <li>✓ ●<sup>5</sup> tick()</li> </ul>	Add cycle function
✓ Test target function     S    S    init()     S    • <sup>S</sup> tick()     S    • <sup>S</sup> alarmForBlind()		Start time: 0 ms Cycle time: 100 ms ✓ ● <sup>S</sup> alarmForBlind() Start time: 0 ms	Remove Up
User code Stub		Cycle time: 1000 ms	Down Set related file

• [Test Info Tab] > [Before/After call code]

E tick()_0 ×					- 8
Test Info (crosswalk/tick_test0)					🌲 🕹 èu
Test Structure				Test Info Edit	
Test structure using a tree view and edit the info Name > • stick() • • salarmForBlind()	In	Out	est.	Edit code to be executed before the function call 1 if ( soundSystem == ON ) {	•
Local Static Variable Parameter/return Before call code After call code					
Auto-generated Variable					- 0
Test Info (crosswalk/tick_test0)					* 12 è
Test Structure Test structure using a tree view and edit the info	ormatior		est	Test Info Edit Edit code to be executed after the function call	
Name > <sup>o</sup> <sup>5</sup> tick()	In	Out		1 }	
of alarmForBlind()     Local Static Variable     Parameter/return					
Before call code After call code					
Auto-generated Variable					

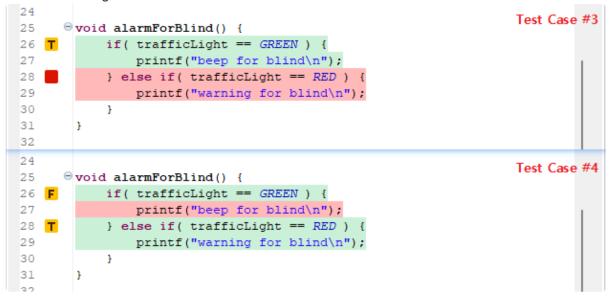
#### **Checking Test Results**

Run the test and verify the results in [Test Case Tab] and [Source Code Editor].

• In [Test Case Tab], confirm that trafficLight changes every 5 seconds.

Fest Case (crosswalk/tick_test0)	#1			Q	E E 🔶 🔶
Parameter	Туре	Input	Expected Value	Host Output	Target Outpu
<ul> <li>trafficLight</li> </ul>	TrafficLightState			0	
<ul> <li>Duration</li> </ul>	unsigned int	5000			
<ul> <li>Total Duration</li> </ul>	unsigned int			5000	
<pre>[est Case (crosswalk/tick_test0)</pre>	#2			Q	E E 🔶 🔿
Parameter	Туре	Input	Expected Value	Host Output	Target Outp
Parameter • trafficLight	Type TrafficLightState	Input	Expected Value	Host Output	Target Outp
Parameter • trafficLight • Duration		Input 5000	Expected Value	Host Output 1	Target Outp

• Use the coverage displayed in [Source Code Editor] to confirm that the signal rang in test case #3 and the warning sounded in test case #4.



### 4. C++ Test Guide

Here's how to test C++ using CT 2023.12.

Guides for C++ Test Using the Class Factory View

### 4.1. Guides for C++ Test Using the Class Factory View

#### **Purpose of using class factories**

When testing C++ source code, it is difficult to test because abstract classes cannot create objects. Class factories can facilitate testing of abstract classes and reduce the iterations that occur when designing class objects.

#### The main features of class factories

- · Automatically create concrete classes that inherits from an abstract class
- · Minimize repetitive tasks by applying them to tests all together

### **Utilizing class factories**

This document explains the basic concepts for testing C++ before using class factories. After that, it explains how to utilize class factories.

- Basic Concept for C++ Test
- Using the Object Creation Code of Abstract Class for Testing
- Design C++ Tests Using Class Factory
- Using Mock Objects in C++ Test

## 4.1.1. Basic Concept for C++ Test

It outlines the basic concepts needed before testing C++ using the Class Factory View.

#### Pure virtual functions and abstract classes

#### **Pure virtual functions**

- Virtual function with declaration but no definition .
- Displayed as = 0.
- · Virtual function implemented in derived class .

#### Abstract classes

- Classes that have pure virtual functions as members.
- Abstract classes cannot create objects.
  - Declare a variable as a pointer or reference type.
    - **ex**. AbstractClass \* class1;.
- Support for polymorphism in object-oriented programming.
- Classes that inherit from an abstract class must override pure virtual functions.
  - If a derived class that inherits from an abstract class does not override a pure virtual function, the derived class is also an abstract class.

```
class Abstract {
       virtual void f() = 0; // pure virtual
}; // "Abstract" is abstract
class Concrete : Abstract {
       void f() override {} // non-pure virtual
       virtual void g(); // non-pure virtual
}; // "Concrete" is non-abstract
class Abstract2 : Concrete {
       void g() override = 0; // pure virtual overrider
}; // "Abstract2" is abstract
int main()
{
        // Abstract a; // Error: abstract class
        Concrete b; // OK
        Abstract& a = b; // OK to reference abstract base
        a.f(); // virtual dispatch to Concrete::f()
        // Abstract2 a2; // Error: abstract class (final overrider of g() is p
ure)
}
```

# 4.1.2. Using the Object Creation Code of Abstract Class for Testing

When analyzing the source code, the object creation code of the concrete class that inherits the abstract class is automatically generated in the class factory so that the object of the abstract class can be created. In the object creation code of the abstract class, a framework for the concrete class is provided so that the user can easily create the concrete class.

When creating a test, if a concrete class that inherits that abstract class exists in the source code, that class is linked with the test, and if the concrete class does not exist, the object creation code in the class factory is linked.

You can apply different types of abstract classes to your tests by adding object creation code.

# 4.1.3. Design C++ Tests Using Class Factory

After Controller Tester 3.5, you can use class factories for most classes, not just abstract classes.

#### Advantages of Controller Tester 3.5 Class Factory

Class factories can be used to reduce simple repetitive tasks.

- · Class objects that get external data
  - Database, external input/output, server, and so on.
- In the case of class objects that need to be designed in a complex way in the Test Editor, but the same should be used for multiple tests.

#### How to create and apply an object using a class factory

- 1. Right-click the class in the Class Factory View and use [Create] to create the class object creation code.
- 2. Modify the class object creation code according to the test design.
- 3. Apply the class object creation code to the tests.
  - Apply all together
  - Apply individually

# 4.1.4. Using Mock Objects in C++ Test

# Purpose of using mock objects

When testing C++ source code, it is sometimes difficult to test because it costs much to create the actual object or the test depends on the object a lot. In such cases, using a mock object that mimics the real object can effectively reduce dependencies on the object. Additionally, you can generate specifications, such as the expected number of calls of the mock to verify that the object is being used as intended.

### Available toolchians

- GCC 6.0 or later
- · Visual Studio 2015 and later

### The main features of a mock object

- · Setting return parameters and return values of a mock object
- · Setting call count for mock object
- · Checking whether the calls occured in a specific order
- · Adding constraints to parameters
- etc

# Mock object usage

This article explains how to use mock objects in C++ tests.

- Creating mock objects
- Generate specifications about mock objects

# 4.1.4.1. Creating mock objects

# **Creating mock objects**

- 1. Open [Test Editor], by double-clicking the test for which to create a mock object.
- 2. In the [Test Info tab], expand the test structure tree and select the object to create a mock.
- 3. Select [Use mock] at the constructor in the test information edit area on the right.

🗄 *CppUnit::TestDecorator::TestDe ×					
Test Info (poco/CppUnit_TestDecorator_TestDecorator_	test0)	)			<b>4</b> è
Test Structure		् 🖽		Test Info Edit	
Test structure using a tree view and edit the information in the test.				Set to create a object and select member field	
Name Test global code	In	Out	^	Constructor	
User code				Use mock	*
Global Variable				User code	
<ul> <li>Test target function</li> </ul>				Use class code	
<ul> <li></li></ul>				Use mock	
Local Static Variable				Function for object return.	
<ul> <li>Parameter/return</li> </ul>					
• est : class CppUnit::Test *					
🗸 🍳 📏 test_mem : class CppUnit::Test					
CS_MOCK_CppUnit_Test					
Before call code					
After call code			×		

# 4.1.4.2. Generate specifications about mock objects

### Generate specifications about mock objects automatically.

- 1. In the Test Information tab, click the mock object that you created.
- 2. In the Test Info Edit area on the right, Click [Generate Sepcification Wizard...] button.
  - If specification about the mock object is empty, [Generation mock specification] wizard automatically appears when you click the mock object.

Generation mock specification	×
Generation mock specification	
① You can specify input and return value or the number of repetitions of function calls. Select the target function to specify the behavior of mock for.	
void Test::run(class CppUnit::TestResult * result)	
int Test::countTestCases()	
std::_cxx11::string Test::toString()	
OK Cancel	
OK Cancel	

- 3. In [Generation mock specification] wizard, select the target function to specify and click [OK] button.
- 4. Edit parameters, return values, and repetitions.

Generation moc	k specification	×
Generation mock	specification	
$\sim$	It and return value and number of function calls for the se you click OK, the mock specification is automatically gene	
Selected function i Mock object name Function:	nformation : csmock_test_mem void Test::run(class CppUnit::TestResult * result)	
Specification settin	g	
Parameter:	run(any value)	Edit parameters
Return value:	no specific return	Edit return value
Repetitions count:	No number of repetitions specified	Edit repetitions
	OK	Cancel

• Click [Edit parameters] button to create a specification of the parameters used by the function.

) Restric	n mock specification ts the input values that can be en g type values must be enclosed ir	ntered as function arguments. When entering a value, character n '' (quotation marks) and "" (double quotation marks).	
Order	Туре	Input value	
1	class CppUnit::TestResult *	any value	
		any value	
		<user input=""></user>	

- Selecting [any value] does not restrict the value of that parameter.
- You can restrict parameter values through [< User input... >]. For example, when you type 1 in the input value and run tests, the test fails if the parameter is not 1.
- Click [Edit return value] button to determine the return value of the function.

Ger Ger	eration mock specification	×
Genera	tion mock specification	
<ol> <li>Sets strin</li> </ol>	the order of return values when function called repeatedly. When entering a value, charact g type values must be enclosed in " (quotation marks) and "" (double quotation marks).	cter or
Order	Return value	Add
1	10	
2	20	Remove
	OK	Cancel

- Select [Add] button to add the value to return when the function is called.
- Select [Remove] button to remove the last added return value.
- If you specify one return value, it will be returned repeatedly.
- When multiple return values are specify, the function returns them in order when called. In this case, the test fails if the function is not called by the corresponding number of return values.
- Click [Edit repetitions] button to create a specification of the number of calls to that function.

Gene	ration mock specification	$\times$
	the number of function repetitions to set.	
-Number value:	of repetitions No number of repetitions specified	_
value.		*
	OK Cancel	

- If you select [No number of repetitions specified], you do not restrict the number of calls.
- Use [< User input... >] to limit the number of function calls. For example, if you set the number of function calls to 3 and run a test, the test fails if the function is not called 3 times.
- [No function call] is the same as specifying a zero number of calls. In this case, the test fails when the function is called.
- To set the return value and the repetitions at the same time, you must write it directly in the Test Editor, referring to the specification you created in [Generation mock specification] wizard.
- 5. Click [OK] button to generate a specification.

### Generate specifications about the mock object yourself

You can modify the specifications created by [Generate Sepcification Wizard...] on Controller Tester or create various specifications yourself. See <u>this document</u> for more information.

# **5. CI/CD Environment and CLI Guide**

Here's how to test in CI/CD environment or in using CLI.

- CT Jenkins plugin Usage Guide
- CLI Guide

# 5.1. CT Jenkins plugin Usage Guide

CT Jenkins plugin is an extension for continuous integration and continuous deployment (CI/CD) of the CT 2023.12 project. By automating tests of CT 2023.12, you can manage your team or organization's development process more efficiently.

# Requirements

#### 1. CT

You must install CT version 2023.12 or higher.

#### 2. Jenkins

For instructions on installing Jenkins, refer to the Jenkins documentation. (Installing Jenkins)

#### 3. CT Jenkins plugin

- Install with hpi file
  - Select and deploy the ct-jenkins-plugin.hpi file in the Deploy plugin item in Manage Jenkins
     Plugins > Advanced settings.
  - 2. Once installation is complete, you can see that the CT environment item has been added to the build environment, and the CT test execution and CT custom command items have been added to Build Steps.

### **Build Environment settings**

Set up the CT execution environment in Manage Jenkins > System > CT (Controller Tester).

#### CT (Controller Tester)

CT Installation Path

(e.g., C:₩Program Files₩Suresoft₩CT 2023)

C:₩Program Files₩Suresoft₩CT 2023

Team Testing Server IP

Team Testing Server Port

CT License

Node-locked

Server OS

Windows

) Linux

IP Port

# **5.1.1. Creating Freestyle Project**

In a Freestyle project, you can configure the project by adding a build environment and build steps. There are two build step options: CT test execution and CT custom command, and it is recommended to use only one of the two build steps to configure the project.

### **Build Steps – CT test execution**

#### **Project Settings**

Set the project you want CT Jenkins plugin to test.

Project Settings ? General Project Team Project Project Name

- General Project
  - For general project, you must enter the path to project exported from CT.
  - When exporting a project, you must include the source code and toolchain.
- Team Project
  - For team project, you must enter the team project name that exists on the Team Testing Server.
  - Team project must be analyzed.

#### **Source Code Settings**

Source code settings are used when regression test with a specific branch in the Git repository. The source code setting synchronizes the source code of the project selected above based on the toplevel path to the source code in the Git repository.

You can check CT test results for source code that changes with this setting in a CI/CD environment. If the option is not selected, the test is performed using the source code of the project selected above.

<	Source Code Settings (for Regression Testing)	?	
---	---	---	--

Git Repository ?

(e.g., https://example.com/user/repo.git)

#### Source Top Level Path

er the top level path for the source code to be tes	ted. Leave this blank if the top level path of the repository is the same as the top level path of the source code.
nch Name	
1	
dentials	
none -	
Add -	

#### **Test Settings**

#### Set test execution options.

Test	Settings
<ul> <li></li> </ul>	Self-healing (Auto Recovery)
	Retry Count ?
	5
	Apply results to the team project (% Caution)
$\Box$	Executing Tests in Linux (RTV)

- Self-healing (Auto Recovery)
  - CT Jenkins plugin automatically performs an integrity check and selects reconfiguration candidates and execute tests.
  - Self-healing runs until all tests succeed or until the number of retries is reached.
- Executing Tests in Linux Select if it is an RTV Project that requires RTV testing.

#### **Report Settings**

Select the format in which you want to generate the resulting report.

### **Build Steps – CT custom command**

This is a build step that allows the user to set workspace and CT CLI settings without using the CT test execution build step.

For detailed usage instructions and issues, please contact us through the technical support contact information at the bottom of manual's <u>troubleshooting page</u>.

# 5.1.2. Creating Pipeline Project

In a pipeline project, you can configure the project by writing a Pipeline script.

# **Build Script Settings**

We recommend using the Snippet Generator in Pipeline Syntax to create pipeline scripts. You can create a script by selecting the build environment setting step (ctEnvironment) and the CT test execution step (ctTestExecution) in the Snippet Generator, and the ctTestExecution step must be included within the ctEnvironment step.

#### Pipeline

Definition

Pipeline script

```
Script ?
```

```
1 * ctEnvironment(ctPath: 'C:\\Program Files\\Suresoft\\CT 2023',
2 ctTestExecution autoCommit: false, credentialsId: '', git
3 }
```

# **Post-build Actions**

In the case of pipeline projects, post-build actions are not added automatically, so they should be added separately.

Post-build actions can also be created using the Snippet Generator, and the steps that can be added are as follows.

#### Archive the artifacts

Select archiveArtifacts in Sample Step and enter the following.

#### Steps

Sample Step

archiveArtifacts: Archive the artifacts

 archiveArtifacts

 Files to archive

 ct/report/TestReport\*.\*, self-healing/\*\*

 Advanced

• Files to archive: ct/report/TestReport\*.\*, self-healing/\*\*

#### Check coverage results

Select ctCoverageReport in Sample Step and enter the following.

#### Steps

Sample Step

ctCoverageReport: Record CT coverage report

```
ctCoverageReport ?
```

Path to xml files (e.g.: \*\*/target/\*\*.xml, \*\*/ct.xml)

ct/report/Jenkins/CoverageResult.xml

• Path to xml files: ct/report/Jenkins/CoverageResult.xml

#### **Check test results**

Select xUnit.Net-v2 in Sample Step and enter the following.

#### Steps

Sample Step

xunit: Publish xUnit test result report
xunit
Report Type
$\equiv$ xUnit.Net-v2 (default)
Includes Pattern
See the list of available jenkins variables as token replacement for this field
Excludes Pattern
See the list of available jenkins variables as token replacement for this field
Skip if there are no test files
Fail the build if test results were not updated this run
Delete temporary JUnit files
Stop and set the build status to failed if there are errors when processing a result file
Add -
Thresholds
Add -

- Report Type: xUnit.Net-v2 (default)
- Includes Pattern: ct/report/Jenkins/TestResult.xml

# Script example

A complete example script based on the above is as follows:

1 = p.	ipeline {
2 3	agent any
	stages {
4 * 5 *	stage('Test') {
6 -	steps {
7 -	steps { ctEnvironment(ctPath: 'C:\\Users\\CodeScroll\\CodeScroll', licenseOption: 'FLOATING', port: '8080', serverIp: '10.10.10.10', serverOs: ';
8 9 10	ctTestExecution autoCommit: false, credentialsId: 'id', gitSourceBranch: 'second', gitSourcePath: 'git@github.com', gitSourceRootPath
9	
	}
11	
12	
13 -	<pre>stage('Publish') {</pre>
14 -	steps {
15	archiveArtifacts artifacts: 'ct/report/TestReport *.*, self-healing/**', followSymlinks: false
16	ctCoverageReport execPattern: 'ct/report/Jenkins/CoverageResult.xml'
17	xunit checksName: ', tools: [XUnitDotNet(excludesPattern: ', pattern: 'ct/report/Jenkins/TestResult.xml', stopProcessingIfError: true) 🔻
18	· ····································
10	F .

# **5.1.3. Check the result**

This explains how to check the results after executing a CT Jenkins plugin project.

### **Project main screen**

You can check test results and code coverage trends on the right side of the main screen. You can check the test report and result files for each self-healing trial in the last successful artifact.

Dashboard > job >		
🗐 Status	Project job	
Changes		🖉 Add description
H Workspace		Disable Project
D Build Now	Last Successful Artifacts	Test Result Trend ← Passed ← Skipped ← Failed 101
Onfigure	TestReport-Merge_original_self-healing.pdf 74.19 KB view     TestReport-Target_original_self-healing.pdf 72.61 KB view	← Passed ← Skipped ← Failed
Delete Project	B TestReport_original_self-healing.pdf 73.12 KB view B 1/CoverageResult.xml 2.46 KB view	16
🖉 Rename	B 1/CoverageResultxml 1.33 KB view B 2/CoverageResultxml 2.46 KB view	14
<b>CT</b> Coverage Trend	2/TestResult.xml 1.69 KB view	10 #2 #3 #4 #5 #6 #7 #8 #9
🔆 Build History trer	B 3/CoverageResultxml     2.46 KB     view       B 3/TestResultxml     1.69 KB     view	Code Coverage Trend
Q Filter builds	<sup>™</sup> → Mathematical Strength and Strength an	80
⊘ #9	B result.json 3.43 KB view	an - MC/DC - branch - cal
Dec 1, 2023, 5:40 PM	Latest Test Result (no failures)	40 Call 20
⊘ #8   <u>Dec 1, 2023, 5:22 PM</u>	Permalinks	د و د و ا
⊘ <u>#7</u>	• Last build (#9), 2 min 33 sec ago	enlarge

### **Build details**

By clicking on a specific build in the build history, you can view detailed information about that build. You can check the results collected in that build, coverage summary, and test results.

Dashboard > job > #9				
🖹 Status	🕢 в	uild #9 (Dec 1, 2023, 5:40:4	44 PM	)
> Changes				
Console Output				
🗹 Edit Build Information	$\bigcirc$	Build Artifacts Build Artifacts TestReport-Merge_original_self-healing.pdf	74.19 KB	view
Delete build '#9'		TestReport-Target_original_self-healing.pdf	72.61 KB	view
		TestReport_original_self-healing.pdf	73.12 KB	view
👼 테스트 자동 복구 결과		1/CoverageResult.xml	2.46 KB 1.33 KB	view view
Coverage Report		1/TestResult.xml 2/CoverageResult.xml	2.46 KB	view
		2/TestResult.xml	1.69 KB	view
Test Result		B 3/CoverageResult.xml	2.46 KB	view
_		B 3/TestResult.xml	1.69 KB	view
← Previous Build		4/CoverageResult.xml	2.46 KB	view
		▲ 4/TestResult.xml	2.05 KB	view
		result.json	3.43 KB	view
		No changes.		
	$\odot$	Started by user		
		CT - Overall Coverage Summary		
		STATEMENT 100%		
		BRANCH 100%		
		CALL 100%		
		Test Result (no failures)		

### Coverage report

You can check the coverage trend graph in more detail by clicking the Coverage Report item on the left sidebar.

Dashboard > job > #5 > CT Coverage						
🖻 Status	CT Coverage	Report				
> Changes	100					
Console Output	80 -	* 				
🗹 Edit Build Information	40					
Delete build '#5'	20	+ statement				
🐖 테스트 자동 복구 결과	0	ę.				
Coverage Report	Overall Coverag	e Summary				
Test Result	name	statement	branch	mcdo	:	call
← Previous Build	all functions	75%	100% M: 0 C: 0	100% M: 0 C: 0	67%	
$\rightarrow$ Next Build			M. 0 C. 0	W. 0 C. 0	W. 1 C. 2	
	Coverage Break	down by Function				
		name	statement	branch	mcdc	call
	fu(signed int, signed	int)	0% M: 2 C: 0	100% M: 0 C: 0	100% M: 0 C: 0	0% M: 1 C: 0
	func1(signed int)		100% M: 0 C: 2	100% M: 0 C: 0	100% M: 0 C: 0	100% M: 0 C: 0
	function4(signed int,	signed int, signed int, signed int)	100% M: 0 C: 2	100% M: 0 C: 0	100% M: 0 C: 0	100% M: 0 C: 1

#### **Test results**

You can check the test results in more detail by clicking the Test Result item in the left sidebar.

Dashboard $>$ test $>$ #8 $>$ Test Results			
🖹 Status	Test Result		
> Changes	15 failures (+15)		
Console Output			6 tests (+86) Took 0 ms.
🗹 Edit Build Information			description
listory	All Failed Tests		
Test Result	Test Name	Duration	Age
← Previous Build			
$\rightarrow$ Next Build	+ .timeOutFunc_test1_1	0 ms	1
	+ .loadFile_test1_1	0 ms	1
	+ .loadFile_test1_2	0 ms	1
	+ .loadFile_test1_3	0 ms	1
	+ .loadFile_test1_5	0 ms	1
	+ .timeout_test1_1	0 ms	1
	+ .exit_error_test1_1	0 ms	1

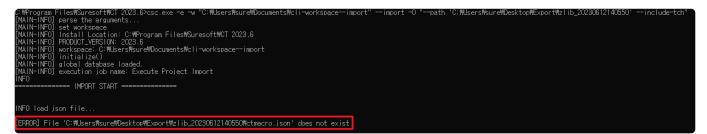
# 5.2. CLI Guide

The scenario explains using CT 2023.12 features by utilizing the Command Line Interface.

• CLI Project Path Reset

# 5.2.1. CLI Project Path Reset

If the source code path during export is different from the source code path when importing, an error occurs as shown below.



In this case, you can specify a new path through mapping while importing the project in the CLI. The mapping method is as follows.

- 1. When you export a project, a PathMappingFile.csv file will be created in the specified directory.
- 2. Edit the PathMappingFile.csv file.
  - Old Path: This is the path specified during the project export process.
  - New Path: This is the path to be used during the import process.
  - status: Indicates the status of the path.
    - If the file path is normal: "OK"
    - If the existing [Old Path] is invalid and no [New Path] has been entered: "The old path is invalid. Please enter a new path."
    - In case the entered [New Path] does not exist: "The new path you entered is not valid. Please check the new path."
    - In cases where source code is included in the export : "This project contains source files. Do not enter new path If you import with source files."
  - a. If the paths on the exporting PC and the importing PC are the same, leave the [New Path] column empty.

	Α	В	C	D
1		Old Path	New Path	status
2	1	C:\Users\sure\Desktop\zlib\22crc32.c		
3	2	C:₩Users₩sure₩Desktop₩zlib₩adler32.c		
4	3	C:\Users\sure\Desktop\zlib\compress.c		
5	4	C:₩Users₩sure₩Desktop₩zlib₩deflate.c		
6	5	C:₩Users₩sure₩Desktop₩zlib₩gzclose.c		
7	6	C:₩Users₩sure₩Desktop₩zlib₩gzlib.c		
8	7	C:₩Users₩sure₩Desktop₩zlib₩gzread.c		
9	8	C:₩Users₩sure₩Desktop₩zlib₩gzwrite.c		
10	9	C:₩Users₩sure₩Desktop₩zlib₩infback.c		
11	10	C:₩Users₩sure₩Desktop₩zlib₩inffast.c		
12	11	C:₩Users₩sure₩Desktop₩zlib₩inflate.c		
13	12	C:₩Users₩sure₩Desktop₩zlib₩inftrees.c		
14	13	C:\Users\sure\Desktop\zlib\trees.c		
15	14	C:\Users\sure\Desktop\zlib\uncompr.c		
16	15	C:\Users\sure\Desktop\zlib\zutil.c		
17				
18				

b. If the code paths on the exporting PC and the importing PC are different, add the path to be used on the importing PC in the [New Path] column.

	Α	В	с	D
1		Old Path	New Path	status
2		1 C:₩Users₩sure₩Desktop₩zlib₩adler32.c	C:\Users\sure\Desktop\newPath\adler32.c	The old path is invalid. Please enter a new path.
3		2 C:\Users\sure\Desktop\zlib\compress.c	C:\Users\sure\Desktop\newPath\adler32.c	The old path is invalid. Please enter a new path.
4		3 C:\Users\sure\Desktop\zlib\crc32.c	C:\Users\sure\Desktop\newPath\crc32.c	The old path is invalid. Please enter a new path.
5		4 C:₩Users₩sure₩Desktop₩zlib₩deflate.c	C:\Users\sure\Desktop\newPath\deflate.c	The old path is invalid. Please enter a new path.
6		5 C:\Users\sure\Desktop\zlib\gzclose.c	C:\Users\sure\Desktop\newPath\gzclose.c	The old path is invalid. Please enter a new path.
7		6 C:₩Users₩sure₩Desktop₩zlib₩gzlib.c	C:\Users\sure\Desktop\newPath\gzlib.c	The old path is invalid. Please enter a new path.
8		7 C:\Users\sure\Desktop\zlib\gzread.c	C:\Users\sure\Desktop\newPath\gzread.c	The old path is invalid. Please enter a new path.
9		8 C:₩Users₩sure₩Desktop₩zlib₩gzwrite.c	C:\Users\sure\Desktop\newPath\gzwrite.c	The old path is invalid. Please enter a new path.
10		9 C:\Users\sure\Desktop\zlib\infback.c	C:\Users\sure\Desktop\newPath\infback.c	The old path is invalid. Please enter a new path.
11	1	0 C:₩Users₩sure₩Desktop₩zlib₩inffast.c	C:#Users#sure#Desktop#newPath#inffast.c	The old path is invalid. Please enter a new path.
12	1	1 C:₩Users₩sure₩Desktop₩zlib₩inflate.c	C:₩Users₩sure₩Desktop₩newPath₩inflate.c	The old path is invalid. Please enter a new path.
13	1	2 C:₩Users₩sure₩Desktop₩zlib₩inftrees.c	C:#Users#sure#Desktop#newPath#inftrees.c	The old path is invalid. Please enter a new path.
14	1	3 C:₩Users₩sure₩Desktop₩zlib₩trees.c	C:₩Users₩sure₩Desktop₩newPath₩trees.c	The old path is invalid. Please enter a new path.
15	1	4 C:₩Users₩sure₩Desktop₩zlib₩uncompr.c	C:#Users#sure#Desktop#newPath#uncompr.c	The old path is invalid. Please enter a new path.
16	1	5 C:₩Users₩sure₩Desktop₩zlib₩zutil.c	C:#Users#sure#Desktop#newPath#zutil.c	The old path is invalid. Please enter a new path.
17				

- 3. Apply the PathMappingFile.csv using the —mapping-file option when importing the project.
  - Example: -e -w "%workSpacePath%" --import -0 "--path '%Project path%' --mapping-file '%PathMappingFile.csv path% --include-tch'"

WUsersWsureWUesktopWExportWzTib_20230612141141WPathMappingFile.csv	include-tch"
[MAIN-INFU] parse the arguments	
[MAIN-INFO] set workspace	
[MAIN-INFO] Install Location: C:WProgram FilesWSuresoftWCT 2023.6	
[MAIN-INFO] PRODUCT_VERSION: 2023.6	
[MAIN-INFO] workspace: C:WUsersWsureWDocumentsWcli-workspaceimport	
[MAIN-INFO] initialize()	
[MAIN-INFO] global database loaded.	
[MAIN-INFO] execution job name: Execute Project Import	
INFO	

4. After the command is executed, the results for each path can be checked in the

#### PathMappingFile.csv file.

	Α	В	с	D
1		Old Path	New Path	status
2	1	C:\Users\sure\Desktop\zlib\adler32.c	C:\Users\sure\Desktop\newPath\adler32.c	ОК
3	2	C:\Users\sure\Desktop\zlib\compress.c	C:\Users\sure\Desktop\newPath\adler32.c	OK
4	3	C:₩Users₩sure₩Desktop₩zlib₩crc32.c	C:\Users\sure\Desktop\newPath\crc32.c	The new path you entered is not valid. Please check the new path.
5	4	C:₩Users₩sure₩Desktop₩zlib₩deflate.c	C:\Users\sure\Desktop\newPath\deflate.c	ОК
6	5	c:₩Users₩sure₩Desktop₩zlib₩gzclose.c	C:\Users\sure\Desktop\newPath\gzclose.c	ОК
7	6	6 C:₩Users₩sure₩Desktop₩zlib₩gzlib.c	C:\Users\sure\Desktop\newPath\gzlib.c	OK
8	7	C:₩Users₩sure₩Desktop₩zlib₩gzread.c	C:\Users\sure\Desktop\newPath\gzread.c	ОК
9	8	B C:₩Users₩sure₩Desktop₩zlib₩gzwrite.c	C:\Users\sure\Desktop\newPath\gzwrite.c	OK
10	9	C:₩Users₩sure₩Desktop₩zlib₩infback.c	C:\Users\sure\Desktop\newPath\infback.c	OK
11	10	C:₩Users₩sure₩Desktop₩zlib₩inffast.c	C:\Users\sure\Desktop\newPath\inffast.c	ОК
12	11	C:\Users\sure\Desktop\zlib\inflate.c	C:\Users\sure\Desktop\newPath\inflate.c	OK
13	12	? C:₩Users₩sure₩Desktop₩zlib₩inftrees.c	C:\Users\sure\Desktop\newPath\inftrees.c	OK
14	13	C:₩Users₩sure₩Desktop₩zlib₩trees.c	C:\Users\sure\Desktop\newPath\trees.c	ОК
15	14	C:₩Users₩sure₩Desktop₩zlib₩uncompr.c	C:\Users\sure\Desktop\newPath\uncompr.c	ОК
16	15	c:₩Users₩sure₩Desktop₩zlib₩zutil.c	C:\Users\sure\Desktop\newPath\zutil.c	ОК
17				

# 6. Test in Real Target Environments

Here's how to test in real target environments using CT 2023.12.

- Target Test Guides
- <u>Debugger User Guides</u>
- Target Build Guide

# 6.1. Target Test Guides

This user guides document describes how to execute target tests using CT 2023.12.

- <u>Texas Instruments Code Composer Studio</u>
- <u>STM32cubeIDE</u>
- <u>Wind River Workbench</u>
- Wind River Workbench

# 6.1.1. Texas Instruments Code Composer Studio

#### 1. Creat a CT 2023.12 project.

File	Edit Search Project Target Window Help				
	New	>	c	C/C++ Project with Source Files	
	Close	Ctrl+W		C/C++ Project from Embedded(CodeWarrior, Green Hills, NEC)	
	Close All	Ctrl+Shift+W	ø	C/C++ Project from Visual Studio Project	
		chi c	<b>B</b>	C/C++ Project from Existing CodeScroll Project	
	Save	Ctrl+S		Create a C/C++ project with CPI File	
B	Save All	Ctrl+Shift+S	1	Create a C/C++ project with Build Information	
	Rename	F2	6	C/C++ Project from RTV Build	
	Refresh	F5	8	C/C++ Project from RTV Build Command	
	Switch Workspace			C/C++ Target Test Project with Source Files	
<u>ک</u> ا	Import		<b>B</b>	C/C++ Target Test Project from Embedded(CodeWarrior, Green Hills, NEC)	
~	Export		2	Create a C/C++ target test project with Build Information	
	· · · · · · · · · · · · · · · · · · ·		83	C/C++ Target Test Project from RTV Build	
	Properties	Alt+Enter	8	C/C++ Target Test Project from RTV Build Command	
	Exit			Other	Ctrl+N

#### 2. Select a created Code Composer Studio toolchain.

Ct New Proje	ect	_		$\times$
New Project				
Create new				
Project name	c CCStest			
Location:	D:\CT34_workspace\201112			
Select Toolc	hain			
Default	Toolchain Name	Description		
	CCS toolchain			
	GCC ARM 7.2			
	gcc5.4			
	gcc7.5			
	GCC 4.7 (32bit)	Automatically gene	rated.	
	GCC 5.3 (32bit)	Automatically gene	rated.	
	Microsoft Visual Studio 2010 (32	Automatically gene	rated.	
	Microsoft Visual Studio 2010 (32	Automatically gene	rated.	
			Toolchain	Setting
	< Back Next >	Finish	Cance	el

3. Select source files to test.

🛃 New Project				×
New Project				
Choose source files				
Top Directory: D:\CCSworkspace\test			Bro	wse
Directory	File			
✓ ✓ test	type filter text			
> Z .settings	main.c			
<ul> <li>Debug</li> <li>dargetConfigs</li> </ul>				
Text file written list of source files				
Text me written list of source mes			Brov	
* The entered text file is separated by	;' or newline characte	er.	Rem	nove
1 items selected				
< Back Ne	ext > Finish		Cance	

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

Codescroll Controller Tester KTV(Rem File Edit Search Project Window	iote Target Verifier) + Target Plugin (x64) Help		
🕞 CCStest		New Project New Test Imp	👱 💆 🗟 🔅 Host 🌒 Target 👱 🐼 Global Search (Ctrl+3) 🔍 🗍 😰 📔
Show Coverage			📄 🗈 Coverage 🚽 MC/DC 🏥 Stub 👸 Class Factory 🐇 Control Flow Graph 🐇 Call Graph 🍃 Function Call Hierarchy 🎦 En
g Test Navigator 📄 😫 🗢 🖻 (		Unit Test Integration Test	
✓ is CCStest 1 ✓ is Default Module 2	2		<ul> <li>Statement Coverage</li> </ul>
> C main.c > C Update_Modify	3 ⊕/** 4 * main.c 5 */ 6 ⊕ int main(void)	Run	(1 / 0 / 0) 1 <b>100.09</b>
	7 { 8 return 0;	Name	Result Coverage
	9 }	✓ ✓ main()	(1 / 0 / 0) 1 100.0% (1/1)
	10	COMPUT	(1/0/0)1
	4	×	
Error 😂			
rrors in the CCStest project.	Dpen log in editor:		
Type filter text Message	Log location Error		
		v	
1 item selected			132M of 478M

5. To use debuggers, set up in Code Composer Studio and CT 2023.12. For more information, refer to <u>Texas Instruments Code Composer Studio</u>, a sub-topic of <u>Controller Tester Debugger User</u> <u>Guides</u> in this document.

# 6.1.2. STM32cubeIDE

1. Setting Target environment

This document describes how to perform target testing using STM32cubeIDE for STM32 family targets.

The application example environment is as follows, and ST-Link debugger is used.

No.	개발 환경(	0S)	빌드 환경	(OS)	개발 언어	통합 개발 환경	경(IDE)		엄파일러	빌드 방식	타깃(실행 환경)	
NO.	종류	버전	종류	버전	개를 진약	조류 종류		종류	버전	(Makefile, IDE)	아키텍처	칩셋(Chipset)
1	Windows	10	Windows	10	с	Stm32cubeide	1.6.1	Arm-none- eabi-gcc	GNU Tools for STM32 9-2020- q2- update.20201001- 1621) 9.3.1 20200408	IDE	ARM Cortex-M7, 32Bit MCU	SMT32F7 Series
2	Windows	10	Windows	10	C++	Stm32cubeide	1.6.1	Arm-none- eabi-g++	GNU Tools for STM32 9-2020- q2- update.20201001- 1621) 9.3.1 20200408	IDE	ARM Cortex-M7, 32Bit MCU	SMT32F7 Series

### Target test application and execution order

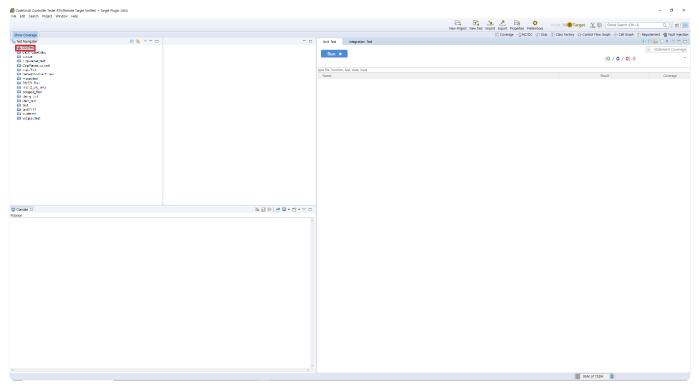
Properties for CCSTest				— 🗆 X				
type filter text	(i) When run	nning the target test, it will overwrite th	ne so filling out the required fields on the					
Inclusion and Exclusion - Info	• <b>1</b> • <b>C</b>	300 + (2 + 14382638; + ida		Import Target Environment Settings				
✓ Target test	Settings							
Target environment set	Analysis	Name	Value					
/ 1650	Build	Toolchain	OC5 TM 032012.6x					
	Run	Status	This tool diam is supported. Cletteccs20eccetcolcecompileetingt v2000 18.12.4.Lisebined					
	etc.	C Compiler						
	010.	System Header(C Compiler)						
		Library(C Compiler)						
		C++ Compiler	C.WtWccs930WccsWtcolsWcompilerWti-	cqt-c2000_10.12.4.LTSMbInVel				
		System Header(C++ Compiler)						
		Library(C++ Compiler)						
		Name:	]					
			]					
		Description:						
< >		L						
			4	Apply and Close Cancel				

- On [right click on project] -> [properties] -> [Target test] -> [Target environment Setting], Just fill out the Property Analysis tab and close it after applying. The target test document is a manual build method, so other tabs do not affect the test.

2. Execute test case unit with [Run Target Test Case]

Unit Test	Integra	ation Test		
Run 🕨				
type file, function,	, test, state	e, issue		
Name				
<pre>✓ ✓ main()</pre>	1			
	] main(). sel 1	_0		
		Сору	Ctrl+	с
		Paste	Ctrl+	V
		Duplicate	Ctrl+	D
		Duplicate multiple times		
	×	Delete	Delet	te
		Rename	F	2
		Add Test Case		
		Host Output Value -> Ex	pected Value	
		Target Output Value -> 8	Expected Value	
	Run Test Case			
	3	Run Target Test Case		
		Add Stub		
	ð.	Inspect Debug Info		

- For accurate testing, run them in test cases.



- If you go through steps 1 and 2, the project of CT 2023.12 will be locked as above.

#### 3. Clean and build the project in STM32cubeIDE

		oject Run Wind							
000	• % • 🗟 × 🙆 🖆 • 6	🔋 • 🖻 • 🚱 •	🎋 • O • 🏊 • 🙋 🤌		• 🗢 🕂 🖆 🕕				Q 时 🖬 🖬
ct Explorer (	n 🖻 😫 🏹	8 - 0 2	acresidad, pociariano que	Barres Barry deletting	🖸 ovdg/ceóbialog: Mitalle/) in óslóbalosi	Salar Samerale	📓 And No, Inc. 2014		Station 2 Schuld Sugar
cm_ma	New		E					-	日本 現代 十十
Binar Inclu	Go Into		THE IN LIKE CODE SECTION	9.**					<ul> <li>Deibilanskoder o SertiFisgild</li> </ul>
C	Open in New Window		[an. ] <sup>1</sup> LONGARD FROM No.	N					<ul> <li>Debtart, storate contracts</li> <li>Sensitive/Ranks (without software)</li> </ul>
👝 In	Show in	Alt+Shift+W	100 H 1 1 1						<ul> <li>Sandivert attributes and edit</li> </ul>
se sr			20. Substant Design Weill Extended Link	diardae estry point.					<ul> <li>SoundRayGrand under conternell</li> </ul>
Drive	Paste	Ctrl+C	pan 11						<ul> <li>Sonciadorel abbulacionel compared Part 2014/1 Source</li> </ul>
FATE	Delete	Ctrl+V Delete-	Notions and a (vol.4)						<ul> <li>dam and the DAA Elect of</li> </ul>
Mida	Source	Deleter	140 77 1503 000 EM	31 L M					<ol> <li>SystemClod_Darlight() - rest</li> </ol>
Touci	Move		2 10 OF 10 OF 10	1.11					O Milly Configuration and A <sup>1</sup> Molectic procession and A <sup>1</sup> M
	Rename	F2	10. RT /* NT Codepose						e Wolf A ht white all
👝 bu			in restorting;						<sup>1</sup> MO, CAC, PRIVAC, 1998
👝 co 🚵			City of the city of the						(i) May be \$20 years with the inter-
≥ 9ª 🛃 ≥ 9			3.5 SOUTH Charles						A <sup>+</sup> MO_ACC_(int)/site matil     A <sup>+</sup> MO_ACC2 (int)/site matil
🕞 S	Build Project		ER Princip brack						H <sup>1</sup> Mathematical Control (1998)
	Clean Project		(iii) (ii						A MOLOW AND AND AND A MARKED AND AND A MARKED AND AND AND AND AND AND AND AND AND AN
-	Ketresh	F5	<ul> <li>Max Configure</li> </ul>						<ul> <li>Holpsoniech (optimities) : A (1212) State (optimities) : A (1212)</li></ul>
ac Mil	Close Project		12						<ul> <li>A statistic provide rules</li> </ul>
Y_1 ap	Close Unrelated Project		<ul> <li>P. Boot of all p</li> <li>M. DB(g)</li> </ul>	prijkerin, bitidien de Sach k	reafies and the Systems "V				e WUW Shickels was
X Ac Debu	Build Configurations		2.8.						
Relea	Build Targets		3.8 P. 1513, 000, 100 57	24 Julia 17					<ul> <li>HOUSER AND MODEL</li> <li>HOUSER AND MODEL</li> </ul>
acm	Index		23 /* ISBN 000-340	Late W					A NO CORTS LATE MARK
acm_	Show in Remote Systems view		to P believe the						(a) Matthe University and
	Run As		[0] SystemClock_LorA [10]	- (i )					e water and an
STM: 🎋	Debug As		TO A MER OVER 100	V there is					Hourseletter     Addition     Additi
	Profile As	4	DE PERSONE DE	London A.					OF MACHINE COMPANY AND
	Team		P.0.						<ul> <li>Subbullueses with</li> </ul>
	Compare With		<ul> <li>M. Setticize all</li> <li>M. Setticize all</li> <li>M. Setticize all</li> </ul>	configuration distances of the					<ul> <li>Foods00(Falords): rec B = 0.065, Second 70, 200</li> </ul>
	Restore from Local History		30 10 108, 34, 310						<ul> <li>Alma Lationity and</li> </ul>
	Generate Code		No. 10.000, 30.0034 No. 10.000, 20.0034 No. 10.000, 20.0034						<ul> <li>Sand Jack-old (1996)</li> </ul>
	Convert to C		DE DE MELLINGTON						() Vel, MAL Peorg, Tracks
~	Run C/C++ Code Analysis Configure		NO HOLMSCHARTS						<ul> <li>H. UKBAR, NEW SKOCK</li> <li>CRE, Jan Jake of the c</li> </ul>
	-		<ul> <li>In project (in)</li> <li>In SPECI S2 (in)</li> </ul>						<ul> <li>Ust Comp Taskheidh : vitil</li> </ul>
	Properties	Alt+Enter	Ser 18 WELSELLD						() Call Milk 264 Selder 1 vold
			NO POINT DATA NO POINT DATA						<ul> <li>East, Remodil (Los) (add) : 20</li> <li>Statistical Security (Security) (Security)</li> </ul>
			IN ROMAN AND	d101:					<ol> <li>RP (VR) Amond Amond Str.</li> </ol>
			THE REPORTS AND T						<ul> <li>Complete disability</li> </ul>
			TO 10 KAVIC AND 3					S	<ul> <li>man(all) finit</li> </ul>
			<					>	¢
		8	Problems 🖉 Tasks 📮 Cor	sole 💱 🥅 Properties	= 🗶 💥 🗟 ⊮ 🧲 🏈	🖳 📑 🖬 🕶 📑 👘 🖬 Bui	ld Analyzer 🚊 Static Stack Analyzer 💡	🔗 Search 🏻 🐁 🕯	Breakpoints 🛶 Progress 🖾 🧏 🖇
		<te< td=""><td>erminated&gt; acm_main Debug  </td><td>STM32 Cortex-M C/C++ Application] ST-LINK</td><td>(ST-LINK GDB server) (Terminated 2021, 7, 12, 오후 2:5)</td><td></td><td>erations to display at this time.</td><td></td><td></td></te<>	erminated> acm_main Debug	STM32 Cortex-M C/C++ Application] ST-LINK	(ST-LINK GDB server) (Terminated 2021, 7, 12, 오후 2:5)		erations to display at this time.		
		Ve	erifying			^			
		ve							
						¥			
		1							

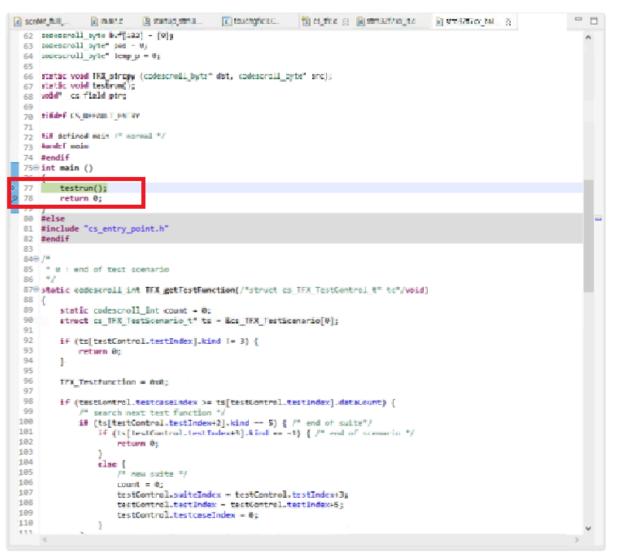
- Clean the exported source in STM32cubeIDE and build it.

#### 4.Debug in STM32cubeIDE

acm_main_Works	Space - acm_main/Core/Src/main. Refactor Navigate Search P	c - STM32CubeIDE	Hele	– a ×
			mmy • O + O₂ + Ø # # = # @ @ # : ∅ + ≅ + ♡ ↔ + ↔ + [#] 0	Q 😰 🖬 🖬
Project Explorer	x E\$7	8	er/Algeboldeness 📓 natus 12 👔 estapate 1377786a 😨 Sudapate 2006esetille de Médicale 📲 sudas 📓 endificadas 📓 etablicadas	- D S Outrie 12 S Suid Broat: - D
✓ mi acm_ma     → mi acm_ma     → mi Binar     → mi Inclu     ✓ mi Core     → mi Core     → mi S     → Sr     → Sr     → St     → St     → St     → St     → St     → St	New Go Into Open in New Window Show In Copy Pathe Delete Source Monu Rename Import Build Project Clean Project	Alt+Shift+W i Ctrl+C Ctrl+V	<pre>No Products core table</pre>	<ul> <li>Experimental processing and the part of t</li></ul>
in ac y⊡ ap in Ac > ⊘ Debu > ⊘ Relea in acm_ acm_	Close Unrelated Project Build Configurations Build Targets Index Show in Remote Systems view Run As		/* Hereit of all peripherals, this defines the flack hour face and the <u>Spinol</u> , "* NAL_STR(); /* uses come ascent <u>puble</u> "/ /* isotoper the spinor for spinor the spinor clock "/ Systemicolog(setting);	<ul> <li>MCAP party starts and</li> <li>MCAP party starts and</li> <li>MCAP party starts and</li> <li>MCAPAP party starts and</li> <li>MCAPAP party starts and</li> <li>MCAPAP party starts and and</li> <li>MCAPAP party starts and and</li> <li>MCAPAP party starts and and</li> <li>MCAPAP party starts and and and and and and and and and and</li></ul>
6-c	Compare With Restore from Local History Generate Code Convert to C Run C/C++ Code Analysis Configure Properties	Alt-Enter	No (NEW, SALE) NO (NEW, SALE) NO (NE CARG) NO (NE CARG)	- Taxata(), religiosity, and     - Fixed a(), religiosity, and     - Solvers, tell checks, and     - Solvers, tell checks, and     - Solvers, tell checks, and     - Cotta Nation, tel
🚰 acm_main		<termi< td=""><td>Jens Console II Properties EX 🙀 📴 Di Di Console II Properties Stack Analyzer 🔌 Static Stack Analyzer 🔌 Static Stack Analyzer 📣 Static Stack Analyzer 👋 Static Stack Analyzer 🖓 Static Stack Analyzer 🖓 Static Stack Analyzer Static Stack Stack Analyzer Static Stack Static Stack Static St</td><td>aarch 💊 Breakpoints 🔤 Progress 🕄 🦉 🔮 😁 🗆</td></termi<>	Jens Console II Properties EX 🙀 📴 Di Di Console II Properties Stack Analyzer 🔌 Static Stack Analyzer 🔌 Static Stack Analyzer 📣 Static Stack Analyzer 👋 Static Stack Analyzer 🖓 Static Stack Analyzer 🖓 Static Stack Analyzer Static Stack Stack Analyzer Static Stack Static Stack Static St	aarch 💊 Breakpoints 🔤 Progress 🕄 🦉 🔮 😁 🗆

- If the build is successful, run debug.

5. Execute after setting a break point in return 0;



- The starting point of the code is main in cs\_tfx.c. Put a break point before 'return 0;', which is the point at which testrun(); ends.

#### 6. Check the log in the ct\_target\_log expression view

		🕼 🚾 🖻 💠 💥 🎉 🗂 🖻
xpression	Туре	Value
s_deif_total_command		Error: Multiple errors reported.# Failed
s_deif_total_command.s_defib_command_		Error: Multiple errors reported.# Failed
(x)= u button input.BIT .setup	unsigned int	0
🐳 Add new expression		

- In the expression view, click Add new expression to add an array containing the log (ct\_target\_log).

kpression	Туре	() 🚾 🖻 ( 💠 🗶 🎉 ( 📑 전 Value
s_deif_total_command		Error: Multiple errors reported.# Failed
s deif total command s defib comman	nd	Error: Multiple errors reported.# Failed
ed= u button input.BIT _setup	unsigned int	
🥬 ct_target_log	char [1000	0x2000020c <ct_target_log></ct_target_log>
> 🖂 [099]	char [100]	0x2000020c <ct_target_log></ct_target_log>
> 🔄 [100199]	char [100]	0x20000270 <ct_target_log+100></ct_target_log+100>
> 🔄 [200299]	char [100]	0x200002d4 <ct_target_log+200></ct_target_log+200>
> 🔄 [300399]	char [100]	0x20000338 <ct_target_log+300></ct_target_log+300>
> 📾 [400499]	char [100]	0x2000039c <ct_target_log+400></ct_target_log+400>
> 🚍 [500599]	char [100]	0x20000400 <ct_target_log+500></ct_target_log+500>
> 🕞 [600699]	char [100]	0x20000464 <ct_target_log+600></ct_target_log+600>
> 🚍 [700799]	char [100]	0x200004c8 <ct_target_log+700></ct_target_log+700>
> 🚍 [800899]	char [100]	0x2000052c <ct_target_log+800></ct_target_log+800>
> 🚍 [900999]	char [100]	0x20000590 <ct_target_log+900></ct_target_log+900>
> 🚍 [10001099]	char [100]	0x200005f4 <ct_target_log+1000></ct_target_log+1000>
> 🚍 [11001199]	char [100]	0x20000658 <ct_target_log+1100></ct_target_log+1100>
> 📑 [12001299]	char [100]	0x200006bc <ct_target_log+1200></ct_target_log+1200>
> 📑 [13001399]	char [100]	0x20000720 <ct_target_log+1300></ct_target_log+1300>
> 🚍 [14001499]	char [100]	0x20000784 <ct_target_log+1400></ct_target_log+1400>
> 📑 [15001599]	char [100]	0x200007e8 <ct_target_log+1500></ct_target_log+1500>
> 📑 [16001699]	char [100]	0x2000084c <ct_target_log+1600></ct_target_log+1600>
> 📑 [17001799]	char [100]	0x200008b0 <ct_target_log+1700></ct_target_log+1700>
> 📑 [18001899]	char [100]	0x20000914 <ct_target_log+1800></ct_target_log+1800>
> 📑 [19001999]	char [100]	0x20000978 <ct_target_log+1900></ct_target_log+1900>

- You can check the contents of ct\_target\_log as above.

7. Check if it ends with	n CSET# (whether of	or not a normal te	est is performed)
--------------------------	---------------------	--------------------	-------------------

		🖾 🚾 E i 🕈 💥 🙀 i 🗂 🖻	
Expression	Туре	Value	1
ct_target_log[169]	char	97 'a'	
et_target_log[170]	char	105 'l'	
<pre>60- ct_target_log[171]</pre>	char	110 'n'	
(+)- ct_target_log[172]	char	44 ','	
(v) ct_target_log[173]	char	49 '1'	
60+ ct_target_log[174]	char	54 '6'	
00+ ct_target_log[175]	char	50 '2'	
(v) ct_target_log[176]	char	54 '6'	
(v)- ct_target_log[177]	char	48 '0'	
(v)- ct_target_log[178]	char	55 '7'	
60- ct_target_log[179]	char	48 '0'	
(v)= ct_target_log[180]	char	57 '9'	
(v)= ct_target_log[181]	char	56 '8'	
(4)= ct_target_log[182]	char	48 '0'	
(v)= ct_target_log[183]	char	62.'>'	
60= ct_target_log[184]	char	67.'C'	
(x)= ct_target_log[185]	char	83 'S'	
(<)= ct_target_log[186]	char	69 'E'	
(x)= ct_target_log[187]	char	84 'T'	
(x)= ct_target_log[188]	char	35 '#'	
(x)= ct_target_log[189]	char	0.40	
(v)= ct_target_log[190]	char	0 .#0.	
<pre>(v)- ct_target_log[191]</pre>	char	0 .#0.	
(i) ct_target_log[192]	char	0 .#0.	
(+)- ct_target_log[193]	char	0 .#0,	
(v) ct_target_log[194]	char	0 '₩0'	
60+ ct tarnet log[195]	char	0 'W0'	

- When the last part of the log ends with CSET#, it can be judged that the test ended normally. Therefore, you can check once whether the test is running normally in the expression view.

8. Add ct\_target\_log to monitor memory in memory view

Console	Problems	Executables	R Debugger Console	e 🚺 Memory 🛛	🔗 Search	🔐 FreeRTOS Task List	FreeRTOS Queues
Monitors			÷	× %			

- Add ct\_target\_log by clicking '+' to Monitors in memory view for memory dump.

×	Monitor Memory						
	Enter address or expression to monitor						
×	ct_target_log						
	OK Cancel						

9. Export from memory view to log path of CT 2023.12 project / Check if the file is normally created in the path

🖸 Console 🛐 Problems 🕥 Executables 👰	Debugger Console	Search 🔐 Fr	HERTOS Task Lis	t if FreeRTOS	5 Queues	
Monitors	🕂 💥 🎉 ct_target_log	: 0x2000020C	<hex> (3)</hex>	🔶 New Render	rings)	
ct_target_log	Address	0 - 3	4 - 7	8 - B	C - F	
	2000020	00000000	D2EA0008	B1010020	43535452	
	2000021	3C383338	30373435	30342C31	37383838	
	2000022	35343935	392C382C	342C3133	3E435354	
	2000023	52234353	53543C61	636D5F6D	61696E2C	
	2000024	31363236	30373039	38303E43	53535423	
	2000025	0A435354	433C3133	32373134	34383934	
	2000026	3436342C	323E4353	5443230A	43534F53	
	2000027	3C317265	7475726E	5661722C	2D31333E	
	2000028	43534F53	230A4353	45533C31	3E435345	
	2000029	53230A43	5354423C	31313131	31313131	

- Click the export button in the memory view to download the memory of ct\_target\_log to a file.

- Format is RAW Binary, Start address is the start address of ct\_target\_log of expression view, and Length specifies the array size of ct\_target\_log. (Even if the length of the log is shorter than Length, it does not affect the test.)

↓ 🔄 🚽 i scan					– 🗆 ×
45 8 34	10				~
	지 않았다. 이 것의 가격 한 이 것 위치 같이 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	。 第二時 17 第1	-) [-] [-] [-] [-] [-] [-] [-] [-] [-] [-]	<ul> <li>101 (121)</li> <li>102)</li> <li>102)</li></ul>	2 B
← → × ↑ 📙 × Ϥ	PC > 트립 디스크	(D.) + Suresoft + CITISQ_WSF	* # acm_maile > .csdata > target >	scan 🗸 🗸	5
# 92.60	^	이름	수정한 날짜	유형 크기	
198 54		scan_result.csv	2021-07-08 오후 3:14		1KB
4 Gecr		targetResult.log	2021-07-12 오후 3:38	텍스트 문서	10KB
N +++1	9				
EE 신설	1				
ACHE1					
ACMU1					
ACMP1					
E DE SACETAR					
📥 Gorffeter					
u na					
🛊 22 - 141					
- Carlos and Carlos an					
B 964					
M 24					
- 118 QM					
<b>M</b> 42					
) #T					
24 Y 8 9 Y 7 KO					
Seconde Dependent D	× (7.1				
김 철말 이 사람 철말 실험을	A PERSON NEWS				

- Check if the file is normally created in the specified path.

#### 10. Restoring source file from CT 2023.12

CodeScroll Controller Tester RTV(Remote Target Verifi Ele Edit Segrch Project Target Window Help			- 8 X
			🔁 🗗 这 🖉 🛱 🌞 Host 🗰 🖉 Target 👱 🐼 (Sichal Search (Cril+3) 🔍 🔢 🔀
Show Coverage			🛐 Coverage - 🖓 MC/DC 👔 Stub 🦉 Class Factory 🚸 Control Flow Graph 👘 Call Graph 💼 Requirement 📲 Fault injectio
Reg Test Navigator	E % V - D	" D Unit Test Integration Test	
CCSTest Cickhouse_test			<ul> <li>Statement Coverage</li> </ul>
cocam		Run 🕨	(0 / 0 / 0) 0
CppParser_test CppParser_vs_test			
CODUTest		type file, function, test, state, issue	
<ul> <li>defaultAbstractClass</li> <li>macrotest</li> </ul>		Name	Result Coverage
POCO Test			
POCO_VS_TEST     poppad_Test			
string_test			
stub_test			
test1111			
Context testmitte vsCppuTest			
- vscppures			
		Restore a source file	×
		There are source files which the target test code has been applied. Restore it?	
		× Ref	
		<ol> <li>You can restore it only if the target test code has been exported as an automatic type.</li> </ol>	
		<ol><li>It should be restored after building a binary for the target test.</li></ol>	
Console 22		No No	
Polarion			
		^	
		v	
<			
EP CCSINK			102M of 732M 🗯 Open Project (2%)

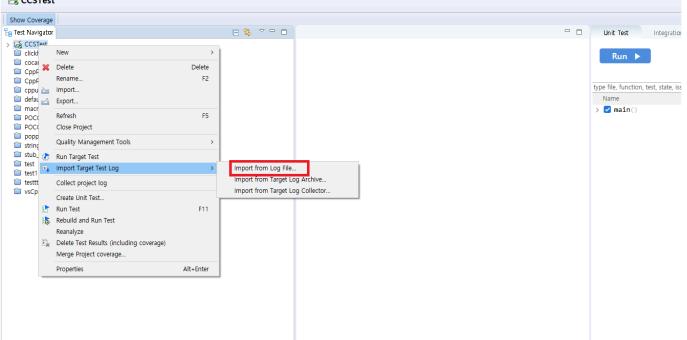
- Restore the source file from CT 2023.12 to get the target test log.

11. Import Target Test Log -> Import from Log File

 Image: CodeScroll Controller Tester RTV(Remote Target Verifier) + Target Plugin (x64)

 File
 Edit
 Search
 Project
 Target
 Window
 Help

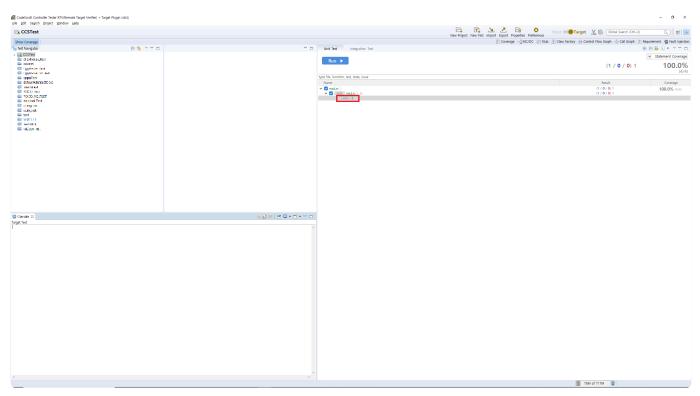
#### 🕞 CCSTest



- Click [Import Target Test Log] -> [Import from Log File]

	CodeScroll Controller Tester RTV(Remote Target Veri Ble Edit Segrch Broject Target Window Help								- ø ×
					New Project New Test	📐 🙇 🗟	🔅 Host 💷 Target 👱 🖁	Global Search (Ctrl+3)	۹ 🖬
	Show Coverage							v Graph 👘 Call Graph 💼 Rer	quirement 📲 Fault Injection
<pre> i i i i i i i i i i i i i i i i i i i</pre>	Pg Test Navigator		Unit Test Integration Test					Œ	🗎 👺 E 🔹 🗢 🗖
	> GS CCSTest								<ul> <li>Statement Coverage</li> </ul>
<pre>     transformed to the final image is a set of the f</pre>	cocam		Run 🕨					(0 / 0 / 0) 0	-
<pre>Bit Control to Co</pre>	CppParser_test								
<pre>B contraint B contraint</pre>	CODUTest								
	i defaultAbstractClass								Coverage
	POCO Test		> 🗹 main()				(8)	0 / 0) 0	
	POCO_VS_TEST								
	poppad_Test								
	stub_test								
	i test								
	i testtittitt								
Control toget for toget	vsCppuTest								
Control toget for toget									
Control toget for toget									
Control toget for toget									
Control toget for toget		Import a target t	est los	×					
							×		
Image: Section of the section of t				.csdata > target > scan	v 8	م	1		
Image: Section of the section of t		1 .x - 47				ii • 🔟 🙆			
Conside 12     Image: Solution 2000     Solution 2000     Solution 2000     Solution 2000       Conside 12     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     Solution 2000     Solution 2000       Solution 2000     Solution 2000     Solution 2000     <				^	수정한 날짜				
Concrete 1:1         Concrete 1:1<									
Constant Reconsta									
Altion     Image: Project and Project an	Console 23								
4 10 - 20       Image: Im	Polarion								
4									
4 4 5 = 32		1 84							
			29-1						
Concentration on py y If the Late of the product to py of the the py of the py		- 4 2 4 (2	¢						
4 15°2 v < v < v < v < v < v < v < v < v < v		- *********	(898(2))						
4 15°2 v < v < v < v < v < v < v < v < v < v		_ Kanning	08.91						
		4 8517							
			v c				>		
			DO 0 BAR tweetRevillion			~			
					L	94M 844			
		~							
	<						145M of 73	2M 💼	

- Import the file created in step 9.



- You can confirm that the test case was successfully executed and the coverage was measured.

# 6.1.3. Wind River Workbench

This document describes how to perform target testing using Wind River Workbench IDE in the VxWorks 6.9 target execution environment.

The application example environment is as follows, and the log interface uses TCP socket communication.

No	No 개발 환경(OS)		빌드 환경(OS).		개발 언어.,	통합 개발 환경(IDE) 컴		컴파일러		빌드 방식	타깃(실행 환경).	
	종류	버전,	종류	버전,		종류,	버전.,	종류	버전,		아키텍처.	칩셋(Chipset).
1.5	Windows.	10.1	VxWorks.1	6.9.1	C.1	Workbench.	3.3.1	GNU.1	PPC85XXe500v2.	Makefile generated by	x86.1	P2020NXN2KFC.
										the IDE.		

As this guide is a target test guide, it is assumed that project creation and analysis have been completed.

Problems that occur in each process of the guide document can be resolved through <u>Controller Tester Target Plug-in Troubleshooting Guide</u>

#### Target test application and execution order

#### 1. Setting Target environment

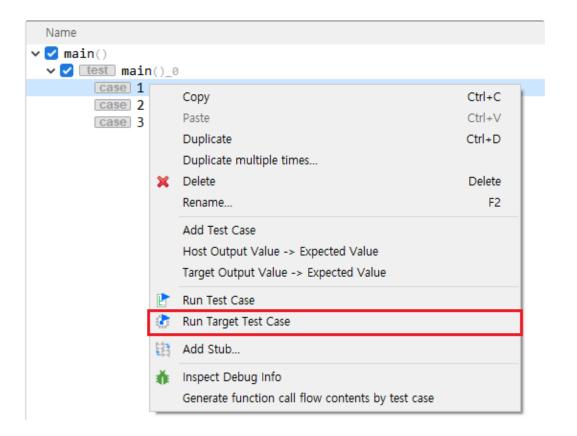
Properties for TESTI 1111				Х				
type filter text	(i) When running the target test, it will overwrite the so filling out the required fields on the Build/Run t	ab. 🤇		<b>₩</b> 00				
Inclusion and Exclusion Info	► WindBiver ► PRC-Disb ► 5.7 ► PowerPC ► modebugger Import Tar	get Envir	onment S	ettings				
<ul> <li>Target test</li> <li>Target environment sett</li> <li>Test</li> </ul>	Settings       Analysis       Batch script:		Brow	se				
	Build       % You must select an execution script (.bat) that will run the project on the target after the         Run       Log interface         etc.	X You must select an execution script (.bat) that will run the project on the target after the build. Log interface						
	socket	Restor	e <u>D</u> efault	S				
	<pre>1//The file is included in all test source files. 2//Include files related to TCP/IP. 3//If the target source does not include the file associated with t 4//The file is used to store target test results. 5// 6//The order of function calls is as follows: 7// 8// int main(){ 9// cs_io_initialize(); 10// while(exist_test){ 11// result = run_test(); 12// cs_io_putbyte(result); 13// cs_io_flush(); 14// } 15// cs_io_finalize(); 16// return 0; 17// } 18 19//Do not modify this header file.</pre>	he sock	et comr					
	20 #include "cs tfx types h" <		>					
< >								
	Apply and Cl	ose	Cancel					

 On [right click on project] -> [properties] -> [Target test] -> [Target environment Setting], Just fill in the log interface of the Run tab, apply and close. The environment covered in this document is a manual build and run method, so the other tabs do not affect testing. The source below is the log interface applied to the example environment (VxWorks 6.9).

```
//Do not modify this header file.
#include "cs tfx types.h"
//Below is an example.
#define AF INET
                       2
#define SOCK STREAM 1
#define htons(x) (x)
struct in addr {
        unsigned int s addr;
};
struct sockaddr in {
        unsigned char sin len;
        unsigned char sin family;
        unsigned short sin port;
        struct in addr sin addr;
        char sin zero[8];
};
int sock;
struct sockaddr in addr;
//This function called at test start.
void cs io initialize()
{
sock = socket(AF INET, SOCK STREAM, 0);
if (sock < 0)
{
 printf("create socket failed\n");
 }
memset (&addr, 0, sizeof(addr));
addr.sin_addr.s_addr = inet_addr("211.116.222.180"); // IP address of the PC
where the CT 2023.12 is installed
addr.sin_port = htons(2019); // The port to be used by the target log collect
or
addr.sin family = AF INET;
addr.sin len = sizeof(addr);
if ((connect(sock, (struct sockaddr_in *)&addr, sizeof(addr))) < 0) {</pre>
```

```
printf("connect failed");
}
else
 printf("connected to server!\n");
printf("Controller Tester Init End!!!!\n");
}
//This function called at test end.
void cs io finalize()
{
//Close socket.
close(sock);
}
void cs io flush()
{
}
//This function prints the test result
void cs_io_putbyte(char v)
{
//Send the target test result.
printf("%c",v);
send(sock, &v, 1, 0);
}
```

#### 2. Target Test Execution

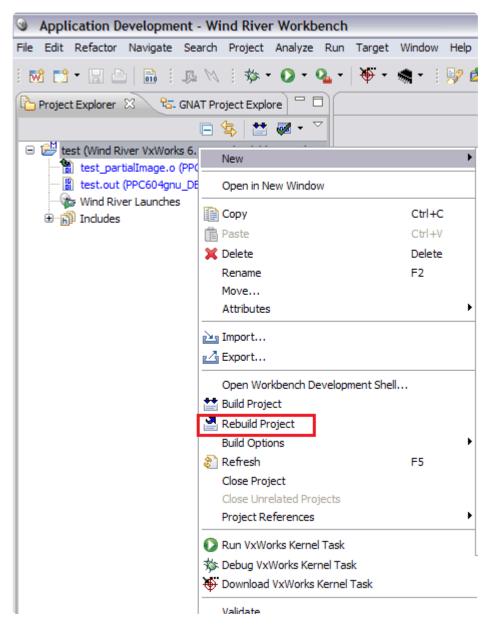


• For accurate testing, execute the test case unit by using the [Run Target Test Case] menu. You can test multiple test cases or functions at once, if memory is available.



• Project in CT 2023.12 will be locked when the target test is executed.

#### 3. Build



• Rebuild Project the exported code in Wind River Workbench IDE.

#### 4. Target log collector settings

- Set the target log collector according to the port created in the log interface and run it.
- The target log collector is installed in %appdata%\CodeScroll\TargetLogCollector\ TargetLogCollector.exe.
- The first time you run the target log collector in cmd, %appdata%\CodeScroll\TargetLogCollector\ setting.ini is created. You can configure the target log collector with this file.
- The example below is the target log collector setting according to the log interface example.

```
[LogReceiveServer]
; TCP, UDP server port
port=2019
; tcp, udp, uart or serial
protocol=tcp
; timeout(second)
timeout=60
```

Suresofttech

```
lastString=CSET#
; serial(UART) port (Windows: COM#, Linux: /dev/ttyS#)
serialPort=COM1
; serial port setting
baudRate=9600
dataBits=8
stopBits=1
parity=0
flowControl=0
[ScanLog]
; log directory(default: scan/log)
dir=
; log file extension(if empty, scan everything)
fileExtension=log
; begin character when filtering the string of log (ascii code with value sepa
rator ;)
beginCharacter=4;5;6
; end character when filtering the string of log (ascii code with value separa
tor ; , 10 is LF)
endCharacter=10
[LogSendServer]
; send log to Controller Tester
port=2020
```

#### 5. Setting the target log collector for CT 2023.12

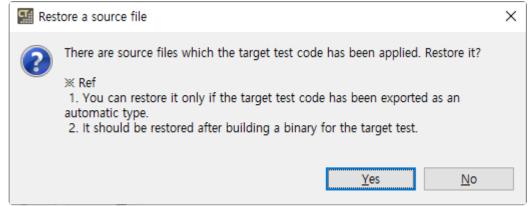
Preferences			— 🗆 X
type filter text	Target Log Collector		<> ▼ ⇒ ₹
<ul> <li>&gt; Analysis</li> <li>Editor</li> <li>Exclusion</li> <li>Language</li> <li>Performance</li> <li>Source File Types</li> <li>&gt; Target Test</li> <li>RTV Server</li> <li>Target Log Collector</li> <li>&gt; Test</li> <li>&gt; ToolChain</li> </ul>	Auto-detect of target lo     Use the default target log     W if check 'Use the default tar     Target log collector info:     Auto-detect cycle(sec): 60	-	Open configuration directory with Controller Tester run automatically.
			Restore <u>D</u> efaults <u>A</u> pply
			Apply and Close Cancel

 In CT 2023.12, in [Preferences] -> [Target Test] -> [Target Log Collector], select Use the default target log collector and turn on auto-detect.

#### 6. Run binary on target

• Move the built binary to VxWorks OS and put it on the target to run it.

#### 7. Restore source from CT 2023.12



• Restore the source from CT 2023.12 to get the target test log.

#### 8. Get target test log

• The created log is automatically loaded according to the set Auto-detect cycle.

### 6.2. Debugger User Guides

This user guides document describes how to use debugger when executing CT 2023.12 target test.

- Lauterbach TRACE32
- PLS Universal Debug Engine
- iSYSTEM winIDEA Debugger
- IAR Embedded Workbench C-SPY Debugger
- <u>Texas Instruments Code Composer Studio</u>
- <u>Microchip MPLAB IDE</u>

## 6.2.1. Lauterbach TRACE32

CT 2023.12 can target test using the TRACE32 debugger.

CT 2023.12 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
- <u>Step1: Setting target environment in CT</u>
- Step2: Run the target test

# 6.2.1.1. Supported target list that can generate cmm script automatically

CT 2023.12 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, mpc5774k, spc574k74, mpc5777m, spc57hm90, mpc574xp, mpc574xg, mpc574xr, mpc577xk, mpc577c, spc570s, mpc5726l, 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

## 6.2.1.2. Step1: Setting target environment in CT

Select Debugger on the target environment setting page of the CT 2023.12. 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

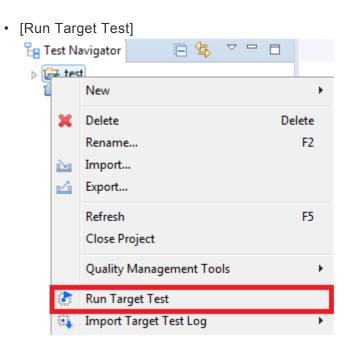
The setting items are displayed according to the selected information. The items you need to set when using the TRACE32 debugger are shown in the table below. Some of the settings are required.

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.

## 6.2.1.3. 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]

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

## 6.2.1.4. 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 CT
- After running Trace32, open the cmm script file (start.cmm) and execute 'debug' (CT\_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
- 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

### 6.2.2. PLS Universal Debug Engine (UDE)

CT 2023.12 can target test using the UDE debugger.

CT 2023.12 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.

CT 2023.12 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 CT
- Step3: Run the target test

## 6.2.2.1. 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.

Select Target Configuration	
Last Used Browse	
Folder to browse :	
{UDE install path}₩Targets₩	<b>_</b>
Additional Filter:	····
Files in folder :	Show descriptions
NXP XPC560P Mini Module with MPC5604P (Jtag)	
Default New Copy	Edit Remove

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

<u>File</u> <u>Edit</u> <u>Debug</u> <u>Show</u> <u>V</u> iews	<u>T</u> ools <u>C</u> onfig	W		
🗅 <u>N</u> ew Workspace	CTRL+N			
Save Workspace CTRL+S				
💐 Save Workspace <u>A</u> s				
Save Workspace as <u>T</u> emplate	<u></u>			
📕 Save <u>V</u> iew Content As	CTRL+ALT+S			
≦ <u>C</u> lose Workspace	ALT+F4			
🖳 Load Program				
Load Program		-		
∖s <u>C</u> onnect Target System				
Sconnect Target System Sconnect Target System	CTRL+P			
<ul> <li>Sconnect Target System</li> <li>∑ Disconnect Target System</li> <li>Print Setup</li> </ul>	CTRL+P			
Sconnect Target System ★ Disconnect Target System Print Setup Print	CTRL+P	►		

3. 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.

FFLAS	H: 512 kByte OnChip	FLASH		<b>–</b>	🔽 Enable	Exit
Index	Start	End	Size	^	Erase	About
0	0x00000000	0x00007FFF	32K			
	0×00000000	0×00000007			Program	Help
	0×00000020	0x000002FF				
	0x00001000	0x000011A7			Verify	General
	0x00002000	0x00007FFF				
1	0x00008000	0x0000BFFF	16K		HW Protect	
	0x00008000	0x0000BFFF				
2	0x0000C000	0x0000FFFF	16K		SW Protect	
	0x0000C000	0x0000FFFF				
3	0×00010000	0x00017FFF	32K		Test Empty	
	0×00010000	0x00013F7B			restempty	
	0x00014000	0x000170D7		~	Info	Program /
Δ	0~00018000	0v0001EEEE	32K	•		Fillyrain/
	ove All Remo	ive Sel.			Setup	Verify Al

See the manual provided by UDE for details.

#### 6.2.2.2. Step2: Setting target environment in СТ

Select Debugger on the Target Environment configuration page of the CT 2023.12. 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.

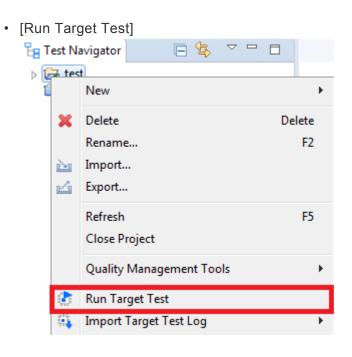
The default scripting language used by CT 2023.12 is visual basic script.

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

### 6.2.2.3. 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]

Unit Test	Integration Test
Run 🕨	1

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.

### 6.2.2.4. 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 CT project
- 3. Verify that the build was successful
- 4. Restore the original source by opening the project in CT
- 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

### 6.2.3. iSYSTEM winIDEA Debugger

CT 2023.12 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 <u>iSYSTEM SDK installation page</u>. 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 CT 2023.12. The iSYSTEM BlueBox iC5000 Unit debugger and NXP's MPC56xx target are used for the examples.

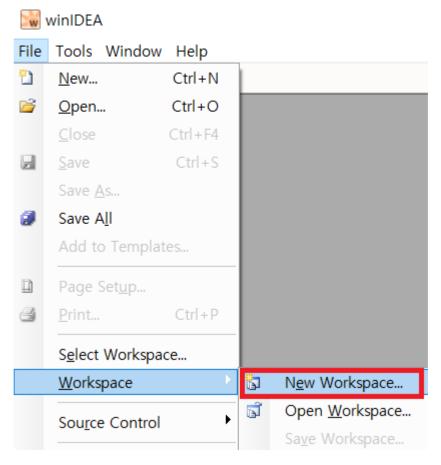
- Preparation for use of iSYSTEM winIDEA
- <u>Step1: Creating and setting up a winIDEA workspace</u>
- <u>Step2: Setting target environment in CT</u>
- Step3: Run the target test

# 6.2.3.1. Preparation for use of iSYSTEM winIDEA

Target testing with winIDEA in CT 2023.12 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 CT 2023.12.

# 6.2.3.2. 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 CT 2023.12 target test.



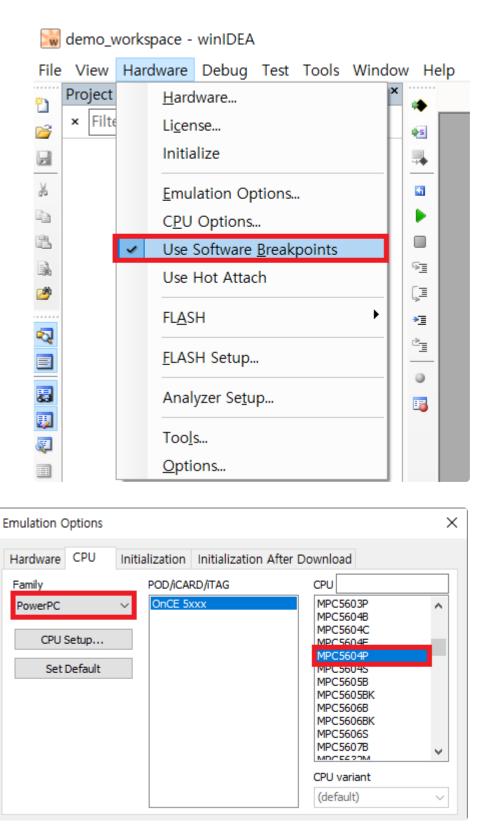
2. First, go to the top menu, select [Hardware]> [Hardware...], and then select the type of the connected BlueBox in the [Hardware Type] tab.

Hardware Configuration	×
Hardware Type Communication	
Hardware	
iC5000, iTAG2k 🗸	
System Configuration	

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.

Hardware Configu	uration			$\times$
Hardware Type	Communic	ation		
(€) USB	Device	BlueBox Unit (serial number) ∨		
⊖ TCP/IP	IP addr	~	Port 5313	
		Use global discovery on UDP port	58371	
) ibridge	Bridge	~	Configure	
	Device	~		
Test	USB: For - Alway: - Avoid			
	- Avoid		ports	

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.

Download	×
Download Files Options Target Download SMP Endian VLE O	verlays
Path Type Offset	
✓       MPC5604P_Example WbinWinternal_FL       ELF	•
New Remove Properties	
Memory area default ~	
Include project output file	
Default file for debugging internal_FLASH.elf	~

6. 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.

🖼 demo_workspace - winIDEA						
File	View Hardwa	re Debug	Test	Tools Window	v Help	)
2	<u>N</u> ew	Ctrl+N		▼□⇔□X	**	
2	<u>O</u> pen	Ctrl+O			•5	
	<u>C</u> lose	Ctrl+F4			<b>.</b>	
	<u>S</u> ave	Ctrl+S			<u></u>	
	Save <u>A</u> s					
۲	Save A <u>I</u> I					
	Add to Temple	ates			®⊒	
D	Page Set <u>u</u> p				(ji	
	Print	Ctrl+P			*3	
_					<u> </u>	
	S <u>e</u> lect Worksp	ace			0	
	<u>W</u> orkspace	•	5	N <u>e</u> w Workspac	e	
	Source Contro	J ►	5	Open <u>W</u> orkspa	ice	_
		~		Sa <u>v</u> e Workspac	:e	
<b>2</b>	<u>F</u> ind In Files			Save Workspac	e <u>T</u> em	plate
	E <u>x</u> it	Alt+F4		Close Wor <u>k</u> spa	ce	

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

## 6.2.3.3. Step2: Setting target environment in CT

Select a debugger in the [New Project] wizard of the target test project or [Target environment settings] of the project properties on CT 2023.12. 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 CT 2023.12.

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 <u>iSYSTEM homepage</u> 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.

#### 6.2.3.4. 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] 🔓 Test Navigator 0 \$ 👂 🔂 test New ۲ × Delete Delete Rename... F2 🔤 Import... 🖾 Export... F5 Refresh Close Project Quality Management Tools Run Target Test 27 Import Target Test Log •
- [Run]

Unit Test	Integration Test
Run 🕨	

\* Target tests cannot be run if winIDEA is running. You must exit winIDEA before running the target test in CT 2023.12.

## 6.2.3.5. 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 CT project
- 3. Verify that the build was successful
- 4. Restore the original source by opening the project in CT
- 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

#### 6.2.4. IAR Embedded Workbench C-SPY Debugger

CT 2023.12 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 <u>IAR website</u>.

To test a target with the IAR Embedded Workbench C-SPY in the CT 2023.12, 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 CT 2023.12 is installed before performing the target test.

The list of debugging probes provided by IAR can be found on the <u>homepage</u>.

- Step1: Creating an IAR embedded workbench project
- Step2: Setting an IAR project
- Step3: Setting target environment in CT
- <u>Step4: Run the target test</u>

# 6.2.4.1. Step1: Creating an IAR embedded workbench project

 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.

Workspace	•	џ×
Debug		~
Files	۰.	•
sample_project - De	~	

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.

-		
Workspace	•	φ×
Debug		~
Files	\$	•
🗆 🌒 sample_project - De	~	
⊨-⊞ 🗟 main.c		
🖵 🖬 🛑 Output		

#### 6.2.4.2. Step2: Setting an IAR project

If you created a project, you need to configure the project to use the C-SPY debugging feature. Rightclick 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.

project"	Cypress	>					
	Epson	>					
	EtaCompute	>					
	Faraday	>					
	Fujitsu	>					
	HDSC	>					
Library Options 2 MISRA-C:2004 MISRA	Hilscher	>					
Target Output Library Configuration Library	Holtek	>					
Processor variant	Infineon	>					
	Lapis	>	BlueNRG	>			
	Linear	>	Other	>			
Device     None	Maxim	>	SPEAr	>			ST STM32F429AG
None	Mediatek	>	STM32F0	>			ST STM32F429AI
O CMSIS-Pack	Microchip	>	STM32F1	>			ST STM32F429BE
Endian mode Floating point settings	Micronas	>	STM32F2	>			ST STM32F429BG
0.0M	Microsemi	>	STM32F3	>			ST STM32F429BI
FP0 None	Milandr	>	STM32F4	>	STM32F401	>	ST STM32F429IE
D registers	MindMotion	>	STM32F7	>	STM32F405	>	ST STM32F429IG
	MitsubishiElectric	>	STM32G0	>	STM32F407	>	ST STM32F429II
	NetSilicon	>	STM32G4	>	STM32F410	>	ST STM32F429NE
DSP Extension	NordicSemi	>	STM32GB	>	STM32F411	>	ST STM32F429NG
Mode Secure	Nuvoton	>	STM32H7	>	STM32F412	>	ST STM32F429NI
	NXP	>	STM32L0	>	STM32F413	>	ST STM32F429VE
	ONSemiconductor	>	STM32L1	>	STM32F415	>	ST STM32F429VG
OK. Cancel	Qorvo	>	STM32L4	>	STM32F417	>	ST STM32F429VI
	Redpine	>	STM32L4+	>	STM32F423	>	ST STM32F429ZE
	Renesas	>	STM32L5	>	STM32F427	>	ST STM32F429ZG
	Samsung	>	STM32MP1	>	STM32F429	>	ST STM32F429ZI
	SiliconLaboratories	>	STM32W	>	STM32F437	>	
	Socle	>	STM32WB	>	STM32F439	>	
	Sonix	>	STR7	>	STM32F446	>	
	ST	>	STR9	>	STM32F469	>	
	Texasinstruments	>		_	STM32F479	>	
	Target       Output       Library Configuration       Library         Processor variant       O Core       ARM7TDMI       Image: Configuration       Image: Config	Epson EtaCompute Faraday Fujitsu HDSC Hilscher Utbrary Options 2 MISRA-C:2004 MISRA Hilscher Hotek Hilscher Hotek Infineon Lapis Linear None OCMSIS-Pack None Floating point settings Microsemi MisubishiElectric NetSilicon NordicSemi MisubishiElectric NetSilicon NordicSemi Nuvoton NXP ONSemiconductor Qorvo Redpine Renesas Samsung SiliconLaboratories Socle Sonix ST	Library Options 2 MISRA-C:2004 MISRA   Library Options 2 MISRA-C:2004 MISRA   Target Output Library Configuration Library   Processor variant Core ARM/7TDMI   O Core None Linear   O Core None Maxim   O CMSIS-Pack None Maxim   Mediatek Microchip   MisubihiElectric MisubihiElectric   Big D registers MisubihiElectric   O EB2 D registers MisubihiElectric   DSP Extension Mode Secure   Mode Secure Nuvoton   OK Cancel Qorvo   Redpine >   Samsung >   Sonix >   Sonix >	Library Options 2       MISRA-C:2004       MISRA         Library Options 2       MISRA-C:2004       MISRA         Target       Output       Library Configuration       Library         Processor variant       Library       Other         O Core       None       Lapis       BlueNRG         O Core       None       Maxim       SPEAr         Mediatek       STM32F0       Microchip       STM32F2         Microsemi       STM32F2       Microsemi       STM32F3         Milandr       STM32F4       MindMotion       STM32F4         MindMotion       STM32F6       NordicSemi       STM32F6         MisubshiElectric       STM32F6       NordicSemi       STM32F6         MidMotion       STM32F6       NordicSemi       STM32F6         MidMotion       STM32F6       NordicSemi       STM32C6         NordicSemi       STM32C6       NordicSemi       STM32C6         DSP Extension       Mode       STM32F6       NordicSemi       STM32L6         NVorton       STM32L6       NSPEX       STM32L6       NSPEX       NordicSemi       STM32L4         Repine       STM32L5       Samurg       STM32WB       Sonix       STM32WB	Library Options 2       MISRA-C:2004       MISRA         Library Options 2       MISRA-C:2004       MISRA         Target       Output       Library Configuration       Library         Processor variant       Holtek       >         Core       ARM7TDMI       Library       BlueNRG         Octore       None       Library       SPEAr         Octore       None       SPEAr       SUMARA         Mediatek       STM32F0       Microchip       STM32F1         Microchip       STM32F2       Micronas       STM32F2         Miandr       STM32F4       MindMotion       STM32F4         MindMotion       STM32F4       MindMotion       STM32F4         MisubishiElectric       STM32F4       MindMotion       STM32F4         MisubishiElectric       STM32F4       NATM32F4       NATM32F4         MisubishiElectric       STM32C6       NATM32F7       NXP         Nuvoton       STM32E4       NXP       NXP       STM32E4       NXP         OK       Cancel       Qorv       STM32L4       Refease       STM32L4       Refease       STM32L4       Refease       STM32L4       SUM2W       SUM2W       SUM32W       SUM32W       SUM32W	Library Options 2       MiSRA-C:2004       MiSRA         Faraday       >         Fujitsu       >         Hischer       >         Core       ARM/TDMI         Occe       None         Occes       Processor variant         Occes       None         Occes       None         Occes       None         Occes       Projecters         Occes       None         Occes       Projecters         Occes       None         Occes       None         Occes       None         Occes       None         Origisters       STM32F401         Microstaria       STM32F4         MisubishiElectric       STM32F402         MisubishiElectric       STM32F402         NXP       STM32F403         NXP       STM32F413         ONSemiconductor       STM32L4       STM32F413	Epon       >         Epon       >         Epon       >         EadCompute       >         Faraday       >         Histher       >         Hotek       >         Horec       Other         Core       None         Corse       None         Processor variant       STM32F4         Other       STM32F3         Microschip       STM32F4         STM32F4       STM32F4         Milandr       STM32F4         Bilse       Projesters         Bilse       Trust2one         Mide Secure       Nird         NordicSemi       STM32F4         STM32F4       STM32F410         Nird       STM32F4         STM32F4       STM32F4         OK       Cancel         OK       Cancel         OK       Cancel         Simazet4       STM32F4

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.

Options for node "sample_pr	oject"	×
_		
Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Setup       Download       Images       Extra Options       Multicore       Plugins         Driver       Images       Run to       Images       Imag	Factory Settings
	OK Cancel	

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.

Options for node "c_proj"		×
Category:		Factory Settings
General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	Setup       Interface       Trace       Breakpoints         Reset	- actory settings
		,
	OK Cancel	

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

## 6.2.4.3. Step3: Setting target environment in CT

Select a debugger in the [New Project] wizard of the target test project or [Target environment settings] of the project properties on CT 2023.12. 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 CT 2023.12.

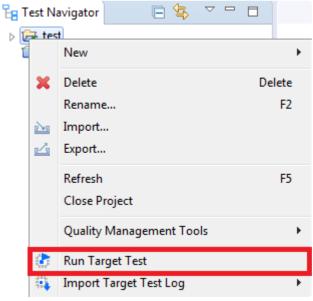
cspy_debug_general_xcl_file_path	The path to the debug.general.xcl file required when using the IAR Embedded Workbench C-SPY debugger. When creating an IAR project, the project file (.ewp) is automatically created in the [setting] folder in the saved location. Required.				
cspy_debug_driver_xcl_file_path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

### 6.2.4.4. 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]



## 6.2.4.5. 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 CT 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

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

CT 2023.12 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.

- <u>Step1: Create a project in Code Composer Studio</u>
- <u>Step2 : Setting target environment in CT</u>
- Step3: Run the target test

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

### 6.2.5.1. 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.

😚 New CCS	Project				×
New CCS P	roject			T	
Create a new	w CCS Project.				
Target:	<select filter="" or="" text="" type=""></select>	✓ TMS320F28335			$\sim$
Connection:	Spectrum Digital XDS560V2 STM	1 USB Emulator	$\sim$	Verify	/

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:	test			
⊡ Use default loc	ation			
Location:	C:₩ccstudio_ws₩	test		Browse
Compiler version:	TI v18.12.3.LTS		~	More
<ul> <li>Tool-chain</li> <li>Project template</li> </ul>	s and examples			
type filter text	· ·	^	Creates an empty project for the selected device. T	he project
Empty P E Empty P E Empty A Empty P		ect	will contain an empty 'ma source-file.	in.c'
<		>		×
			election of example project projects for selected device	
?	< Back	1	Next > Finish	Cancel

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

CT 2023.12 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.

# 6.2.5.2. Step2 : Setting target environment in CT

- 1. Creat a CT 2023.12 project. For more information to create the project, refer to <u>Texas Instruments</u> <u>Code Composer Studio</u> in this document.
- 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 CT 2023.12. 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.

Properties for CCStest				— 🗆 X		
type filter text (1) Whe	n running the targe	t test, it will build the te	est colly by filling out the required fields	s on the Run tab. 🛛 🖕 👻 🔫 🖛	,	
1110	C2000 ► 6.2 ►	TMS320F28x 🕨 ide		Import Target Environment Setting	<u>qs</u>	
✓ Target test Target environment set	Use build	script				
> Test Analy:	Name		Value			
Run	use_stdio_h	eader	false			
etc.	use_std_stri		false			
	ide_director		C:\ccs1011\ccs			
	workspace		D:₩CCSworkspace			
	project_nan		test			
	toolchain_k	Ind	ti			
	Name:	ide_directory_path	Value: C:\ccs1011\ccs	Browse		
	Description:	The path to the directory where Code Composer Studio ide is installed				
< >			]			
				Apply and Close Cancel		

#### • Fileds 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 CT 2023.12

5. Enter needed informations on [Run] tab of [Target environment settings] for running target tests.

CI Properties for CCS					— 🗆 X
type filter text	When runn	ing the targe	et test, it will build ar	nd run the test code.	← -
Inclusion and Exclusion Info	► TI► C	2000 🕨 6.2 🌶	► TMS320F28x ► ide		Import Target Environment Settings
✓ Target test	Settings				
Target environment set	Analysis Build Run etc.	Name ccxml_path target_binal debug_prot cpu_name Name: Description:	ry_path	Value C:#Users#SURE#workspace_v9#CCSTe C:#Users#SURE#workspace_v9#CCSTe *	2 2
< >			case sensitive.		
				[	Apply and Close Cancel

#### 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</i> \targetConfig\ <i>target-name</i> .ccxml
target_binary_path	Enter a path of binary file created during build in Code Composer Studio. ex) <i>project-path</i> \Debug\ <i>project-name</i> .out
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</i> <i>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)

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

Debug		
	debug_probe	cpu_name
Device	Spectrum Digital XDS560V2 STM US	3 Emulator C28xx

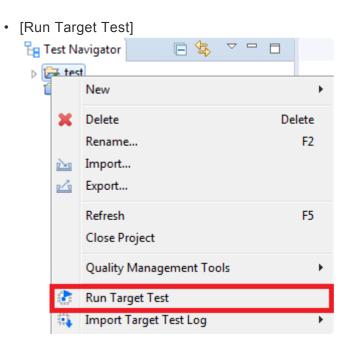
When only one debugger is connected to the target, debug\_probe can be left as the

default (\*). For single core cpu, you do not need to set cpu\_name.

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

### 6.2.5.3. 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]



If Code Composer Studio is running during target test execution, a compilation error occurs.

For more information on debug scripting in CCS, see the Texas Instruments home page.

### 6.2.5.4. 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.
- 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

## 6.2.6. Microchip MPLAB IDE

This document describes how to run target tests using the Microchip MPLAB IDE in three steps.

- <u>Step1: Debugger script settings</u>
- Step2: Setting target environment in CT
- Step3: Run the target test

### 6.2.6.1. Step1: Debugger script settings

In order to perform the target test in CT 2023.12, the mdb.bat file included in MPLAB must be modified so that the log output from the debugger can be saved in a file format. The mdb.bat file path is as follows.

#### For windows 32 bit

• C:\Program Files\Microchip\MPLABX\vn.nn\mplab\_ide\bin\mdb.bat

#### For windows 64 bit

• C:\Program Files (x86)\Microchip\MPLABX\vn.nn\mplab ide\bin\mdb.bat

Modify the code in the last line of the mdb.bat file as follows.

#### before modification

"%jdkhome:exe =exe%" -Dfile.encoding=UTF-8 -jar "%mdb\_jar%" %1

#### after modification

```
call "%jdkhome:exe =exe%" -Dfile.encoding=UTF-8 -jar "%mdb_jar%" %1 >> %CT_TAR
GET PATH%\mdb log.txt
```

Microchip MPLAB has a Korean encoding issue, so you should not include Korean in the CT 2023.12 workspace or project name.

# 6.2.6.2. Step2: Setting target environment in CT

Select the debugger in the target test project creation wizard in CT 2023.12 or in the target environment settings in the project properties. The list of debuggers to choose from depends on the toolchain selected when creating the project.

Set the debugger to ide.

Microchip ► XC16 ► others ► MPLAB-PIC ► ide

Setting items are displayed according to the selected information. Required fields are displayed in red in CT 2023.12.

ide_directory_path	The path to the directory where Mplab ide is installed. Required.
project_directory_path	The directory path of the project. Required.
make_path	The path to the make.exe file. Just enter the path to make.exe used when building in the mplab project. Required.
Dia	any file to be unloaded to the target (binary location generated during build)

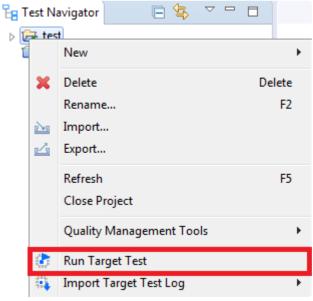
target_binary_path	Binary file to be uploaded to the target (binary location generated during build). Required.
debugger_tool	You can select the debugger tool information (select among ICD3, RealICE, PICkit3, SIM, PM3, LicensedDebugger, LicensedProgrammer, SK). Required.
chip	Product name of the chip under test (exdsPIC33EP512MU814). Required.

In order to perform the target tests in CT 2023.12, the mdb.bat file must be modified as in <u>Step1</u>. When the target environment setting is finished, click the [OK] or [Finish] button. You are ready to perform target testing.

### 6.2.6.3. 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 clicking the [Run] button in the Test View.

• [Run Target Test]



• [Run]



## 6.3. Target Build Guide

CT 2023.12 guides you through building target test code using target project information.

- IAR Embedded Workbench IDE
- <u>Texas Instruments Code Composer Studio</u>
- <u>CodeWarrior IDE</u>
- Hightec Development Platform IDE
- Tasking VX IDE
- <u>Renesas CS+ IDE</u>
- MPLAB X IDE
- <u>Microsoft Visual Studio</u>
- GNU Compiler

### 6.3.1. 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 of target that can be selected from Core of Processor variant
- · Build tab

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

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'.



General Options			
Static Analysis			
Runtime Checking	Library Options 2	MISRA-C:2004	MISRA-C:1998
C/C++ Compiler Assembler			
Output Converter	Target Outpu	( Library Configuration	Library Options 1
Custom Build	Library:	Description:	
Build Actions	Full V	Use the full configuration o	f the C/C++ runtime
Linker	1 un	library. Full locale interface,	C locale, file
Debugger Simulator		descriptor support, multiby	tes in printf and scanf,
CADI		and hex floats in strtod.	
CMSIS DAP	Configuration file:		
GDB Server	\$TOOLKIT_DIR\$\minc\mathcal{Bit}	WDLib Config Full h	
I-jet	STOOLKIT_DIKSTIIICT	C#DLID_COINIg_Fuil.II	
J-Link/J-Trace TI Stellaris	Enable thread suppo	rt in library	
Nu-Link	Library low-level inter	face implementation	CMSIS
PE micro	None	stdout/stderr	
ST-LINK			Use CMSIS
Third-Party Driver	O Semihosted	Via semihosting	DSP library
TI MSP-FET TI XDS	○ IAR breakpoint	🔿 Via SWO	,
11 /03			

### 6.3.2. 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.

Properties for CCStest					_		×
type filter text	(i) When rur	nning the targe	t test, it will build the te	st colly by filling out the required fields (	on the Run tab.	<pre></pre>	⇒ ▼ ▼
Inclusion and Exclusion	► TI ► C2	2000 🕨 6.2 🕨	TMS320F28x ► ide	Import Target Er	nvironmer	nt Settings	
<ul> <li>Target test</li> <li>Target environment set</li> <li>Test</li> </ul>	Settings Analysis	Use build	script				
> Test	Build	Name		Value			
	Run etc.	use_stdio_he use_std_strir	ng_header	false false			
		ide_director workspace		C:\cs1011\cs D:\CCSworkspace			
		project_nam toolchain_ki		test ti			
		Name: Description:	ide_directory_path	Value: C:\ccs1011\ccs		Br	owse
		Description.	The path to the directory where Code Composer Studio ide is installed				
< >>							
				4	Apply and Close	Can	cel

#### Build tab

ide_directory_path	h Directory path of Code Composer Studio <i>ex.C:\ti\ccs930</i>						
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.

### 6.3.3. 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 <i>ex.</i> C:\Program Files (x86)\Freescale\CW for MPC55xx and MPC56xx 2.10, C:\Freescale\CW MCU
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'.

### 6.3.4. 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 ex. C:\HIGHTEC\toolchains\arm\v4.6.5.0
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'.

# 6.3.5. 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 <i>ex. 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'.

### 6.3.6. 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</i> <i>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

	The Preserve of Standard			
	4 (17)100 1			X X
T. Project) 240709 1100 Mt (Microcontrollier) Rin Configurator (Design Tool)	<ul> <li>Bohap Minnakan</li> <li>Ali debap Minnakan</li> <li>Optimization</li> </ul>	TestAdl to Set-assentits and object (Rolds 2		
Code Dana dos (Davigs Tool)	Parlary set minutes	No(Chanland)(p2)		
Cole Canador (Design Tool) Calification (Select Tool) MICH SITUATION (Select Tool)	<ul> <li>Addisonal reliade paths</li> </ul>	Additional include paths[7]		
Program Analyzes (Analyze Tool) The	5 System Indiada padha 9 Record defension	Symen instants sufficial Noon defector (0)		
Contap Intela	<ul> <li>Same and the set of the set of</li></ul>	Race webfinite[1]		
a sector	the standard startup costine therefored seem country, standard library	Yes/Kernal Na		
	RDNps for area Voing standard startsproving	100 cDLod		
	See Edwary Une standard ideary	No.		
	Use standard FO Roary assembli Reating solet data Use multipliar	No.		
	> tempeterdet företen > Ressage	Tails divided Brake(3)		
	factore     force			
	Additional index addit Samba in additional inclusion and a state complete. If Na spline control, only the special Values ("second registers") Compare functions			File Live Pr
o for Popular P.   Bit forume	Chert S Chertal Peter Consta  07/vet Not  97/vet Not  03 (o	NuthSource Philement Procedures of the Press	Note Start - Marting Star Diver	PR Jumple Function
th(One path	per one line): 😱 oll.ut.embedded₩RenesasCSPlusR oll.ut.embedded₩RenesasCSPlusR	.78_2₩TestFixture .78_2₩TestFixture₩cs		
com.codescr	oll.ut.embedded₩RenesasCSPlusR	.78 2₩TestFixture		
th(One path com.codescr com.codescr Browse Permit non- Include sub	oll.ut.embedded₩RenesasCSPlusR	.78_2₩TestFixture .78_2₩TestFixture₩cs		
th(One path com.codescr com.codescr Browse Permit non-	roll.ut.embedded₩RenesasCSPlusR roll.ut.embedded₩RenesasCSPlusR -existent path	.78_2₩TestFixture .78_2₩TestFixture₩cs		
th(One path com.codescr com.codescr Browse Permit non- Include sub aceholder: laceholder ctiveProjectI ctiveProjectI	oll.ut.embedded₩RenesasCSPlusR oll.ut.embedded₩RenesasCSPlusR -existent path ofolders automatically Value Dir C:₩Users₩SURE₩Docum Name TEST	.78_2₩TestFixture .78_2₩TestFixture₩cs		
th(One path com.codescr com.codescr Browse Permit non- Include sub aceholder:	oll.ut.embedded₩RenesasCSPlusR oll.ut.embedded₩RenesasCSPlusR -existent path ofolders automatically Value Dir C:₩Users₩SURE₩Docum Name TEST me DefaultBuild ir C:₩Users₩SURE₩Docum	.78_2₩TestFixture .78_2₩TestFixture₩cs		

# 6.3.7. 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 <i>ex.</i> C:\Program Files (x86)\Microchip\MPLABX\v5.35
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'.

## 6.3.8. 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	<b>ctory_path</b> Installation path of Microsoft Visual Studio <i>ex.</i> C:\Program Files (x86)\Microsoft Visual Studio 10.0	
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'.

### 6.3.9. 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'.

### 7. Identifying the Cause of a Test Error

Occasional errors occur when performing tests on CT 2023.12. At this time, the user can find out the cause of the test error by checking the debug information of CT 2023.12.

### If test execution fails

Progress Information	
Run Test	
Cannot proceed test. Please check test environment(ex:Dll path).	
	Cancel

Debug information verification can be performed even if the test fails. When you execute the [Inspect Debug Info] of the generated test case, the stack trace is displayed in the [Inspect Debug Info], and you can know where the test failed.

### If the result contains errors after running the test

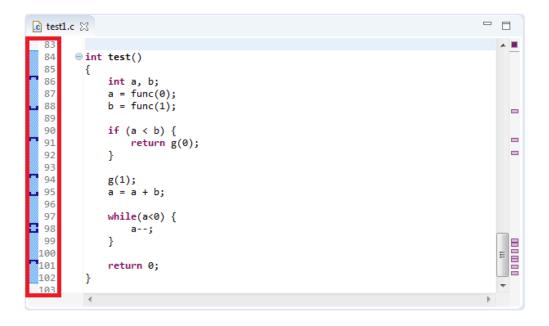
In some cases, after performing a test in CT 2023.12, error results such as Signaled and Abnormal Exit are displayed. When you execute the [Inspect Debug Info] of the failed test case, the function call stack trace is displayed in the [Debug Information View]. If you added a variable/expression to debug, a list of executed variables/expressions is also displayed.

🍺 Debug Information 🔀			(×)= 🗆 🗖
Fest Case (test/func_test0) #1			
Test Case (test/func_test0) #1 Stack trace: ▼ ♣ [Project: test] test run	List of variable/expression: Variable/Expression p (test1.c: 40)	Value 12319812	Location before line

The Stack trace indicates the order of the function calls. The location where the function was called is recorded, and the last execution location is recorded at the top of the Stack trace.

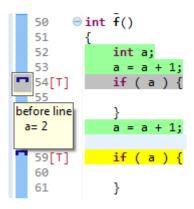
The List of variable/expression represents the variable/expression values executed with the test case.

The list of variables/expressions added to the entire source code can be checked in [List of Variable/ Expressiont] 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 [Inspect Debug Information] in the CT 2023.12 document.

### 8. Virtual Address Usage Guide

You can set the memory for testing the embedded environment by setting the virtual memory address.

1.	Top menu	[Window]	>	[Preferences]	>	[Unit	Test	>	[Virtual	Address	>	[Add]	Selection

Preferences		—	
type filter text	Virtual Address		<
<ul> <li>Analysis         <ul> <li>Analysis</li> <li>Editor</li> <li>Exclusion</li> <li>Language</li> <li>Performance</li> <li>Source File Types</li> <li>Target Test</li> <li>Test</li> <li>Coverage</li> <li>Export test data</li> <li>External editor</li> <li>Perspective</li> <li>Type Partition</li> <li>Unit Test View</li> <li>Virtual Address</li> <li>Theme (CS Labs)</li> </ul> </li> </ul>	Configuration n	ist:	Add Edit(E) Remove
> ToolChain	Begin	End	
		Apply and Close	Cancel

2. After entering the name and range of the virtual address, click the [Add(A)] button

🖫 Virtual me	emory address Cre	ate	×			
Virtual mem	Virtual memory address Configuration					
(i) The count The virtual	of virtual memor memory address	y address value list is up to 50. value format is hexadecimal.				
Basic Config	uration					
Name:	REG_BASE					
Description:			~			
			~			
Virtual mem	ory address Config	guration				
0xFFE40000		- 0xFFE40100	Add( <u>A</u> )			
Begin		End	Delete( <u>R</u> )			
		OK	Cancel			

3. Right-click the project [Properties] > [Unit Test] > [Virtual Address] and select the registered virtual address range in the [Configuration Name] combo box

Properties for Virtual memor	y example	- D X
type filter text	Virtual Address	← → ⇒ ▼
Inclusion and Exclusion Info Target test Contemport Exclusion of coverage External Command Generate test cases aut Run Test Virtual Address	Configuration name:       REG_BASE         Detail information       REG_BASE         Begin       End         0xFFE40000       0xFFE401	00
		~
< >		
		Apply and Close Cancel

4. Using a macro to set a value to a virtual address in [Before call code] of the test structure editor

Test Info (Virtual memory example/testMe\_test0)

Test structure using a tree view and edit the information in the test.		< ⊞ ⊟	Edit and to be succeeded before the forestime call
-			Edit code to be executed before the function call
Name	In	Out	1{
Test global code			<pre>2 int base_addr_value = CS_INT_INPUT(int, "base_addr_value"); 3 CS ADDR SET(int,0xFFE40000,base addr value);</pre>
User code			<pre>3 CS_ADDR_SET(int,0xFFE40000,base_addr_value); 4 }</pre>
Global Variable			- J
<ul> <li>Test target function</li> </ul>			
✓ ● <sup>T</sup> testMe()			
Local Static Variable			
<ul> <li>Parameter/return</li> </ul>			
o <sup>R</sup> returnVar : int		$\checkmark$	
<ul> <li>Before call code</li> </ul>			
base_addr_value : int	$\checkmark$		
After call code			
User code			
✓ Stub			
Host			
Target			
			< > >

**\*** For details about macro, please refer to the <u>Test Maco</u> page in User Manual

5. Edit test case values

<u>2</u>2

JUN	Case (Virtual memory example/tes	the_testoy #1				<	~
arar	meter	Туре	Input	Expected Value	Host Output	Target Output	
	Test global code						
	User code						
	Global Variable						
/	Test target function						
~	<ul> <li></li></ul>						
	Local Static Variable						
	> Parameter/return						
	<ul> <li>Before call code</li> </ul>						
	<ul> <li>base_addr_value</li> </ul>	int	10				
	After call code						
	User code						
-	Stub						

Test Info Test Case Test Code Configuration

## 9. Navigate Source Codes

CT 2023.12 provides shortcuts and context menu in Source Code Editor for user convenience.

### Shortcuts

Item	Shortcut	Description
Open Include Browser	Ctrl + Alt + I	Display the include relationship of the selected file in the [Include Browser View].
Show outline	Ctrl+0	Show outline of selected file in outline popup.
Toggle Source/ Header	Ctrl + Tab	Toggle source file and header file.
Open type in Hierarchy	Ctrl + Alt + H	Display hierarchy of the selected item in [Call Hierarchy View]. (Funtion/Global Variable)
Toggle Mark Occurrences	Alt + Shift + O	Turns the mark occurrence on/off for the item that is positioned by cursor or is specified by block.
Open Declaration	F3,Ctrl+Click	Move to the declaration of the selected item or open the file if it is an include file.
Open Resource	Ctrl + Shift + R	Open a file by searching by name.
References	Ctrl + Shift + G	Display reference to selected item in Search View.
Forward/ Backward history	Alt + Right / Left	Move editor history forward/backward.
Find Next/ Previous	Ctrl+K/Ctrl+Shift +K	Search the selected text forward/backward in the current file.
Toggle Folding	Ctrl + Numpad_Divide	Show/Hide folding icon.
Zoom Out/In	Ctrl + -/Ctrl + Shift + =	Zoom out/in source code editer.
Expand/ Collaspe	Ctrl + Numpad_Add/Nu mpad_Subtract	Expand/collapse the item on the cursor.
Move Line Down/UP	Alt+↓/↑	Move line down/up.
Copy/Duplicate Lines	Ctrl + Alt + ↓ / ↑	Copy lines down/up.

### Context menu

2)	Undo Revert File Save Open With	Ctrl+Z Ctrl+S		
	Show In	Alt+Shift+W >	6	Project Explorer
	Cut Copy Paste	Ctrl+X Ctrl+C <b>Ctrl+V</b>	H 10 10 1	Outline Navigator System Explorer
	Source	Alt+Shift+S >		Properties
	Search Text	>		
다 (x)= 605	Create Stub Add variable/expression to debug View a list of variable/expression to debug			
	Preferences			

ltem	Description
Outline	[Display the outline of the current file in [Outline View].
System Explorer	Open the current file location in Windows Explorer.

🖉 Undo	Ctrl+Z		
Revert File			
Save	Ctrl+S		
Open With	>		
Show In	Alt+Shift+W >		
Cut	Ctrl+X		
Сору	Ctrl+C		
Paste	Ctrl+V		
Source	Alt+Shift+S >		
Search Text	>	Workspace	Ctrl+Alt
Treate Stub		Project	
Add variable/expression to debug		File	
View a list of variable/expression to debu	g	Working Set	
Preferences			

Item	Description
Search Text	Search the selected character string in the target (workspace/project/file) and display it in [Search View].