



3.7 — Last update: Jun 09, 2022

Suresofttech

Table of Contents

1. Target Test Guides	2
1.1. Texas Instruments Code Composer Studio	3
1.2. STM32cubeIDE	5
2. Debugger User Guides	14
2.1. Lauterbach TRACE32	15
2.1.1. Supported target list that can generate cmm script automatically	16
2.1.2. Step1: Setting target environment in Controller Tester	17
2.1.3. Step2: Run the target test	18
2.1.4. Debug the target test	19
2.2. PLS Universal Debug Engine (UDE)	20
2.2.1. Step1: Create a workspace in UDE IDE	21
2.2.2. Step2: Setting target environment in Controller Tester	23
2.2.3. Step3: Run the target test	24
2.2.4. Debug the target test	25
2.3. iSYSTEM winIDEA Debugger	26
2.3.1. Preparation for use of iSYSTEM winIDEA	27
2.3.2. Step1: Creating and setting up a winIDEA workspace	28
2.3.3. Step2: Setting target environment in Controller Tester	33
2.3.4. Step3: Run the target test	34
2.3.5. Debug the target test	35
2.4. IAR Embedded Workbench C-SPY Debugger	36
2.4.1. Step1: Creating an IAR embedded workbench project	37
2.4.2. Step2: Setting an IAR project	38
2.4.3. Step3: Setting target environment in Controller Tester	
2.4.4. Step4: Run the target test	
2.4.5. Debug the target test	
2.5. Texas Instruments Code Composer Studio (CCS v4 and later)	
2.5.1. Step1: Create a project in Code Composer Studio	
2.5.2. Step2 : Setting target environment in Controller Tester	
2.5.3. Step3: Run the target test	
2.5.4. Debug the target test	
2.6. Microchip MPLAB IDE	
2.6.1. Step1: Debugger script settings	
2.6.2. Step2: Setting target environment in Controller Tester	
2.6.3. Step3: Run the target test	55
3. Target Build Guide	56
3.1. IAR Embedded Workbench IDE	57
3.2. Texas Instruments Code Composer Studio	59
3.3. CodeWarrior IDE	61
3.4. Hightec Development Platform IDE	62
3.5. Tasking VX IDE	63
3.6. Renesas CS+ IDE	64

3.7. MPLAB X IDE	
3.8. Microsoft Visual Studio	
3.9. GNU Compiler	68
4. Sharing Projects with Other Users	69
4.1. (Ver.3.3 or later) Guide to Share Projects	
4.1.1. Export project	71
4.1.2. Import project	73
4.2. (Ver.3.2 or earlier) Guide to Share RTV Projects	
4.2.1. Project sharing scenario	
4.2.2. RTV server user guide	
5. Identifying the Cause of a Test Error	
6. Source Code Modification and Test Reconfiguration	
6.1. Run [Test Reconfiguration]	
6.2. In Cases of Detected Modification Automatically	
6.3. In Cases of Undetected Modification Automatically	
7. Navigate Source Codes	
8. Guides for C++ Test Using the Class Factory View	
8.1. Basic Concept for C++ Test	100
8.2. Using the Object Creation Code of Abstract Class for Testing	101
8.3. Design C++ Tests Using Class Factory	
8.4. Using Mock Objects in C++ Test	103
8.4.1. Creating mock objects	
8.4.2. Generate specifications about mock objects	105
9. Virtual Address Usage Guide	108
10. Guides to Import Coverages	112
10.1. Import Coverages by Version	113
10.2. Import Coverages by Conditional Operation Option	114
10.3. Import Coverages by Coverage Type	115

1. Target Test Guides

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

- <u>Texas Instruments Code Composer Studio</u>
- <u>STM32cubeIDE</u>

1.1. Texas Instruments Code Composer Studio

1. Creat a CodeScroll Controller Tester project.

File	Edit Search Project	Target Windov	v He	elp				
	New	>	¢	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				
	Course		B	C/C++ Project from Existing CodeScroll Project				
	Save	Ctrl+S		Create a C/C++ project with CPI File				
G	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				
è	Import		F	C/C++ Target Test Project from Embedded(CodeWarrior, Green Hills, NEC)				
4	Export			Create a C/C++ target test project with Build Information				
			83	C/C++ Target Test Project from RTV Build				
	Properties	Alt+Enter		C/C++ Target Test Project from RTV Build Command				
	Exit		P	Other	Ctrl+N			

2. Select a created Code Composer Studio toolchain.

_	w Project						_		×
	Project e new projec	t							
Projec	t name:	CCStest							
Locati	on:	D:₩CT34_	worksp	ace₩201112					
Selec	t Toolchain								
Defa	ault Too	lchain Name	^		Descrip	tion			
	CCS	toolchain							
	GCC	C ARM 7.2							
	gcc	5.4							
	gcc	7.5							
	GCC	C 4.7 (32bit)			Automa	atically g	generate	d.	
	GCC	C 5.3 (32bit)			Automa	atically g	generate	d.	
	Mic	rosoft Visual	Studio	2010 (32		atically g			
	Mic	rosoft Visual	Studio	2010 (32	Automa	atically g	jenerate	d.	
									_
							T	oolchain	Setting
		< Bac	k	Next >		Finish		Cance	A

3. Select source files to test.

Ci New Project		_		\times
New Project				
Choose source files				
Top Directory: D:\CCSworkspace\test			Brow	wse
Directory	File			
✓ ✓ test	type filter text			
 Settings Debug targetConfigs 	☐ main.c			
Text file written list of source files				
			Brov	vse
* The entered text file is separated by ';	or newline characte	er.	Rem	nove
1 items selected				
< Back Ne	xt > Finish		Cance	1

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

CCStest	🛱 🖬 🖄 💆 🗟	🔅 Host 🔵 🗉 Target 👱 💀 Global Search (Ctrl+3) 🔍 😰 🗖
	New Project New Test Import Export Properties Pro	reterences
Show Coverage		C 🔃 Stub 👸 Class Factory 🚯 Control Flow Graph 🐁 Call Graph 🍃 Function Call Hierarchy 🎦 Err
Test Navigator 😑 😫 🔍 🗖 🗋 main.c 😒	Unit Test Integration Test	
	Run 🕨	 Statement Coverage
3 ● /** > j_ main.c 3 ● j_* main.c 5 % Update_Modify 6 € int main(void)		(1 / 0 / 0) 1 100.0%
7 { 8 return 0;	type file, function, test, state, issue Name	Result Coverage
9 }	vame	(1 / 0 / 0) 1 100.0% (1/1)
10	Comel 1.	(1 / 0 / 0) 1
	v	
C	>	
Tror 🔀		
Type filter text Open log in editor :		
Message Log location Error		
] 1 item selected	v	182M of 478M

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

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(개발 환경(OS)		빌드 환경(OS)		통합 개발 환경(IDE)		지지 통합 개발 환경(IDE) 컴파일러		발 환경(IDE) 컴파일러		빌드 방식	타깃	(실행 환경)
NO.	종류	버전	종류	버전	개발 언어	종류	버전	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

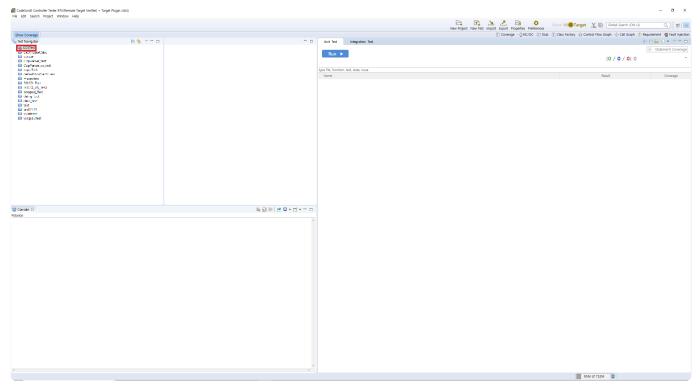
Properties for CCSTest			— 🗆 X
type filter text	(i) When run	nning the target test, it will overwrite th	ne so filling out the required fields on the Build/Run tab. $ \ \ \ \ \ \ \ \ \ \ \ \ \ $
Inclusion and Exclusion		800 + C2 + 143824380 + Ma	Import Target Environment Settings
 Target test Target environment set 	Settings		
> Test	Analysis	Name	Value
,	Build	Toolchain	OCS TM 9320726x
	Run	Status	This toolcham is supported.
	etc.	C Compiler	C.et.#ccs920#ccs#toolcecompile#tillingtilc2000_18.12.4.LISebinect.
		System Header(C Compiler)	
		Library(C Compiler)	
		C++ Compiler	C:WtWccs930WccsWtoolsWcompflerWtFcqt-c2000_10.12.4LTCWoInVel
		System Header(C++ Compiler)	
		Library(C++ Compiler)	
		Name:	
		Description:	
< >			
			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				
✓ ✓ main()	• • •			
V V test) main(). ise] 1	_0		
		Сору	Ctrl	+C
		Paste	Ctrl	+V
		Duplicate	Ctrl	+D
		Duplicate multiple times		
	×	Delete	Dele	ete
		Rename		F2
		Add Test Case		
		Host Output Value -> Ex	pected Value	
		Target Output Value ->	Expected Value	
		Run Test Case		
	٢	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 Controller Tester will be locked as above.

3. Clean and build the project in STM32cubeIDE

plorer 8			acres fall population	an America S & other day	The debut states	antibulies in cases and the state of the	Distance in a	C deb Vio le proc	1.12	Synthetics of Shield Largers
na	New		26	1. A 1914 19				-		EARX+
ar			THE A LER COLL	to cash and						 Detbilantekinder orgentika
104	Go Into		an Charles of the							 Dytection, storage compared
re In	Open in New Window		No. 1							 Sendershade (strend)
Sr	Show In	Alt+Shift+W	Design of the second se	to application entry point.						 SandStatLabilities condi- s SandReiCent under oder
St 🗈	Сору	Ctrl+C.	Control 1							 Sonciación del concerción
ve ns	Paste	Ctrl+V	period and mole (vol.4)							 canned : RAC(308AA); Sat
dd 🗙	Delete	Delete	14C 1	E BROND V						 dm_unt_n: DAA, Rent_0
	Source		Per Las de	CONTRACT DAY						 SystemClod_Damight() are A* MRL Conflictment and
A¢	Move		2 IB /* 1201,000	8 BAD 1, 17						A MORE DANCE THE
	Rename	F2:	And I am look	Incention						+ VOLDAA, MARKET HEL
b.	Import		in instantig)							() MOLONG, HEROLG, NON
	Export		Contraction (Contraction)	Zide-		17 N				()* MU(AA220,
0			La souperate							A ⁺ V0 A00 (inf)/db midi A ⁺ V0 A002 (inf)/db midi
s	Build Project		Ex Plantin b	- Cache		17				H ¹ M3 (Mit anti-only work
	Clean Project		30 00.0x83-00							é Majanéséjantak ing
	Refresh	F5	tal /* star Contra	inenter-						H ¹ M0(S04/AC1(30)(H0)(H0)) (1 ¹ M0(S01)(H0)(H0)(H0)(H0)
ac	Close Project		125	-						HI MALLAN & INCLUDE THE
ap	Close Unrelated Project		[25] P. Bart M. R. M. D.B.(3)	all pripherits bittidies th	Then interface and the Gate	15 V				e ⁺ MUUVTSjindyelde volet
A¢	Build Configurations		25							HANDLIGHT DATE MARKEN HANDLIGHT
bu ea	Build Targets		15 /* ISBN 000	C 1003 (132 17						 HOUSE CARLINGS HOUSE CARLINGS
n	Index		28 /* #SEX 000	C SALLAR V						O ROUGHTS MET MARK
	Show in Remote Systems view			e de range clock 17						(1) Martin Landsonia, 1997.
0	Run As		[31] System lock;							Wayse package and
*	Debug As		No. At ISSN 240	Property and the second s						Organization (Comparison) Organization Organizatio
	Profile As		278.							6 Machine Management
	Team		DE PERSON	C MERCHARK OF						 A SPEAK DARKS with
	Compare With			an all configuration blocks in N						 Text#20(Televal*): rest
	Restore from Local History		[38] 10, 893, 363 (2) 10, 993, 36, 5							 B. S. S. S. SAMON AND G. SAMAR Just Scale (1996)
۵.	Generate Code		No. 10,00,30,0 No. 10,000,20	64.						+ Sound Late-off include
8-c	Convert to C		NA POINT DA	6						 Vet.04.VOL2eorg.74-4
	Run C/C++ Code Analysis		NO ROAD CASE							H URBOAL WILL BADDO
	Configure		10 10 BRCSC							 CRE_Join_Jateroid(): NRC 44 Uat Come Trainethr: Vitil
	Properties	Alt+Enter	16 10 SPECT S 36 10 SPECT SU							() Call Dills 2nd Sales Lott
_			340 PD_0414_04	6#()):						 Lat, Renotle, Just, addp. 1
			200 100000000 200 100000000							e * SDAA, _ Maitzrien, Sequer
			THE REPORTED							 BROYELANOSACIAS Construction Tradición
			200 10 (00471) /# 200 10 (04710 /#							 Invanisation (int)
			C C C C C C C C C C C C C C C C C C C	The share of the					>	<
		6	Problems 2 Tasks	Console 🕴 🥅 Properties		x 🔌 🖹 🖬 🖗 🕼 🖉 -	📑 💌 🗖 📄 🕞 Built	d Analyzer al Static Stack Analyzer	Search .	Breakpoints R Progress 23
				Debug [STM32 Cortex-M C/C++ Applicati					y scalen -0	areautourus -D. reduces 12 36 5
				read format contextua cice + Abblicati	and as more (as much one server) (renn		^ No ope	rations to display at this time.		
		1	erifying							
							~			

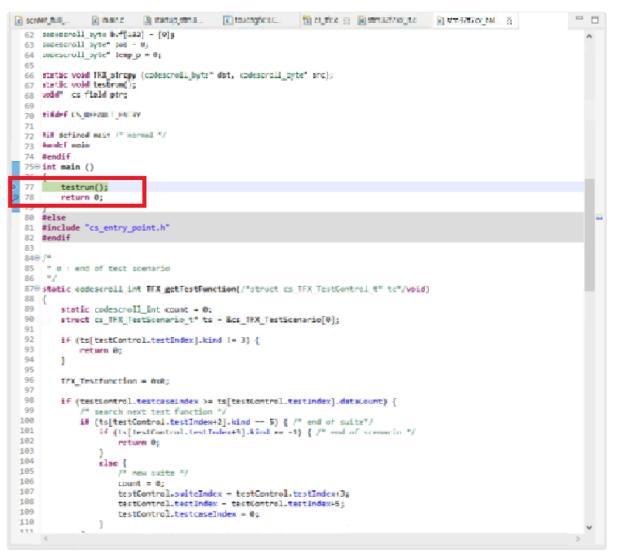
- Clean the exported source in STM32cubeIDE and build it.

4.Debug in STM32cubeIDE

	Space - acm_main/Core/Src/main.or Refactor Navigate Search Pi		– ¤ ×
	· · · · · · · · · · ·	• 63 • 62 • 63 • 1 \$\$ • 0 • 94 • 1 \$\$ 10 11 12 • 51 • 55 \$\$ • \$\$ • [1] [1]	Q 🔡 🖬 🖬
Project Explorer	x 😑 🕏 🏹	🖓 🕴 🗢 🗋 😰 szeregál jegeteteleneze 📓 máraz 12 📓 szelegyés 1977bba 🛒 középészőtő lésszellikébe) Aladitásási – 🇌 szelez 📓 máraz 12 📓 szelegyészőtő lésszellikébe) Aladitásási – 🖏 szelez 📓 máraz 12 📓 szelez jesz 1	= D 💱 Oxtine 😒 🛞 Build Degets 👘 D
	Paste Delete Source Move Rename Import	Abt-Shirty - Version - The regulation -	Provide a second s
> 20 gu > 20 sc > 20 ta > 20 ta ac ac ac ac ac ac ac ac ac ac ac ac ac	Build Project Clean Project Refresh Close Project Close Unrelated Project Build Configurations Build Targets Index	<pre>15 StilletBilletTil</pre>	 M. ALADSE Insert Liver M. ALADSE Insert Liver M. ALADSE INSERT LIVE AND ALADSE INSERT LIVER AND ALADSE INSERT A
	Show in Remote Systems view Run As Debug As Profile As Team	<pre>/ is setigate the space clock */ is graphed book (of the space clock */ is graphed book (of the space clock */ is / is clock clock the space clock */ is / is / is clock the space clock */ is / is</pre>	 A KOWATE LAM - although void A KOWATE LAM - although void A KOWATE LAW - although void
\$.	Compare With Restore from Local History Generate Code Convert to C Run C/C++ Code Analysis Configure Properties	Alt-conter 35	 Transmit University und Exploring Transmitty of the second s
s_ acm_main		215 TV, LawY,	Crucia Master Freuerikord over workersee and the second over the seco

- 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.

Check the log in the ct_target_log expression view

		🕼 🚾 🖻 🕈 💥 🐮 🗂 🗂
pression	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).

			🕼 🚾 🖻 💠 💥 🙀 📑 🖻	
Expression		Туре	Value	
s deif total o	ommand	-	Error: Multiple errors reported.# Failed	
	ommand.s_defib_command		Error: Multiple errors reported.# Failed	
ed⊨ u button int		unsigned int		1
🗸 🥭 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	1	char [100]	0x200002d4 <ct_target_log+200></ct_target_log+200>	
> 📑 [300399	1	char [100]	0x20000338 <ct_target_log+300></ct_target_log+300>	
> 🚍 [400499	1	char [100]	0x2000039c <ct_target_log+400></ct_target_log+400>	Г
> 🚍 [500599]	char [100]	0x20000400 <ct_target_log+500></ct_target_log+500>	Ľ
> 🚍 [600699	1	char [100]	0x20000464 <ct_target_log+600></ct_target_log+600>	L
> 🚍 [700799	1	char [100]	0x200004c8 <ct_target_log+700></ct_target_log+700>	L
> 🚍 [800899	1	char [100]	0x2000052c <ct_target_log+800></ct_target_log+800>	
> 🚍 [900999	1	char [100]	0x20000590 <ct_target_log+900></ct_target_log+900>	
> 📑 [100010	99]	char [100]	0x200005f4 <ct_target_log+1000></ct_target_log+1000>	
> 🚍 [110011	99]	char [100]	0x20000658 <ct_target_log+1100></ct_target_log+1100>	L
> 📑 [120012	99]	char [100]	0x200006bc <ct_target_log+1200></ct_target_log+1200>	
> 🚍 [130013	99]	char [100]	0x20000720 <ct_target_log+1300></ct_target_log+1300>	
> 📑 [140014	99]	char [100]	0x20000784 <ct_target_log+1400></ct_target_log+1400>	
> 📑 [150015	99]	char [100]	0x200007e8 <ct_target_log+1500></ct_target_log+1500>	
> 📑 [160016	99]	char [100]	0x2000084c <ct_target_log+1600></ct_target_log+1600>	
> 📑 [170017	99]	char [100]	0x200008b0 <ct_target_log+1700></ct_target_log+1700>	
> 📑 [180018	99]	char [100]	0x20000914 <ct_target_log+1800></ct_target_log+1800>	
> 📑 [190019	99]	char [100]	0x20000978 <ct_target_log+1900></ct_target_log+1900>	
	001	also fit 0.01	avagagagade with treast law, apage	

- 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)
--------------------------	---------------------	--------------------	-------------------

		約 🚾 日 (🌩 🗶 🎉 (🗂 🖻	000
Expression	Туре	Value	1
et_target_log[169]	char	97 'a'	
ct_target_log[170]	char	105 'l'	
ct_target_log[171]	char	110 in'	
(+)- ct_target_log[172]	char	44 ','	
(v) ct_target_log[173]	char	49 '1'	
(v) 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'	
(v)= ct_target_log[179]	char	48 '0'	
(v)= ct_target_log[180]	char	57 '9'	
(v)= ct_target_log[181]	char	56 '8'	
(x)= ct_target_log[182]	char	48 '0'	
(x)= ct_target_log[183]	char	62.'>'	
64= ct_target_log[184]	char	67.'C'	
(x)= ct_target_log[185]	char	83 'S'	
(x)= 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 10	
(+)= ct_target_log[190]	char	0 .#0.	
et_target_log[191]	char	0 .#0.	
60- ct_target_log[192]	char	0 "₩O'	
(+)+ ct_target_log[193]	char	0 '₩0'	
(+)+ ct_target_log[194]	char	0 .440,	
64. ct target 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	Memory S	🛛 🛷 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 express	on to monitor.
ct_target_log	~
? Ок	Cancel

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

Monitors	💠 💥 🍇 ct_target_log	: 0x2000020C	<hex> 23</hex>	🔶 New Render	ings	
ct_target_log	Address	0 - 3	4 - 7	8 - B	C - F	
	20000200	00000000	D2EA0008	B1010020	43535452	
	20000210	3C383338	30373435	30342C31	37383838	
	20000220	35343935	392C382C	342C3133	3E435354	
	20000230	52234353	53543C61	636D5F6D	61696E2C	
	20000240	31363236	30373039	38303E43	53535423	
	20000250	0A435354	433C3133	32373134	34383934	
	20000260	3436342C	323E4353	5443230A	43534F53	
	20000270	3C317265	7475726E	5661722C	2D31333E	
	20000280	43534F53	230A4353	45533C31	3E435345	
	20000290	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.)

📑 🛃 🥃 🖛 scan					– 🗆 ×
35 E 38 37					^ ()
	18-40 C	- <mark>↓ </mark> X - 2 2 3 3 2 3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	a service and the second by the	2
← → ~ ↑ □ > 4 m → 4	정 디스크	(D.) + Suresoft + CITISG_WSP +	acırı_main > .csdata > target > s	scan v Ö	
× # 92.60	^	이름	수정한 날짜	유형 크기	
198 24	4	scan_result.csv	2021-07-08 오후 3:14	Microsoft Excel 1KB	
4 Decr	2	targetResult.log	2021-07-12 오후 3:38	텍스트 문서 10KB	
Active Active Active Active Active Active Construction Constructin Construction Constructin Const					
5 M 24					
S 🚾 1981 🖼 🖉					
2 📧 42					
5 🔰 #4					
1 Mar 7 B 417 7 KD					
 	~				
12 24 1 21 24 292 4744					

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

10. Restoring source file from Controller Tester

CodeScroll Controller Tester RTV(Remote Target Verifier) + Target I	t Plugin (x64)				- a ×
Elle Edit Segrch Project Target Window Help				Fig. 노. 스. Fig. 부 Host = ●Target 또 🔞 (dobal Search (Ctrl+3))	ت ا
				New Project New Test Import Export Properties Preferences	
Show Coverage Tg Test Navigator	E 😵 🗵 🗉 🖬		Unit Test Integration Test		
Costed		0	unit tex integration tex.		
Contest			Run 🕨		 Statement Coverage
CopParser_test				(0 / 0 / 0) 0	-
CooParser vs test			type file, function, test, state, issue		
cppuTest defaultAbstractClass			Name	Result	Coverage
macrotest POCO_Test					
POCO_VS_TEST					
poppad_Test string_test					
i stub test					
test test1111					
i testiititt					
vsCppuTest					
		Restore a source file	×		
		There are source	te files which the target test code has been applied. Restore it?		
		1. You can re-	tore it only if the target test code has been exported as an restored after building a binary for the target test.		
		2. It should be	restored after building a binary for the target test.		
Console 23		N 51 10	Yes No		
Polarion					
		~			
4		v			
<		· · · · · · · · · · · · · · · · · · ·		102M of 732M	
				- Upen Project. (2.6)	

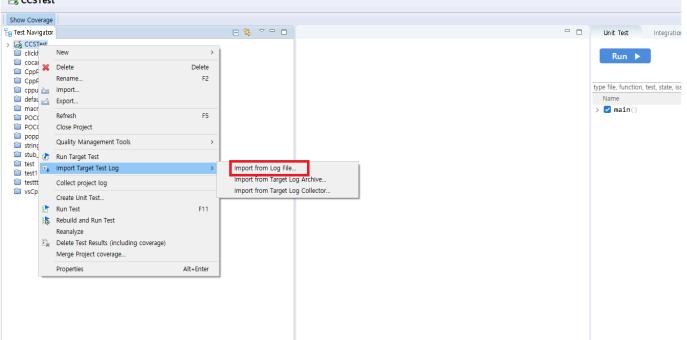
- Restore the source file from Controller Tester to get the target test log.

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

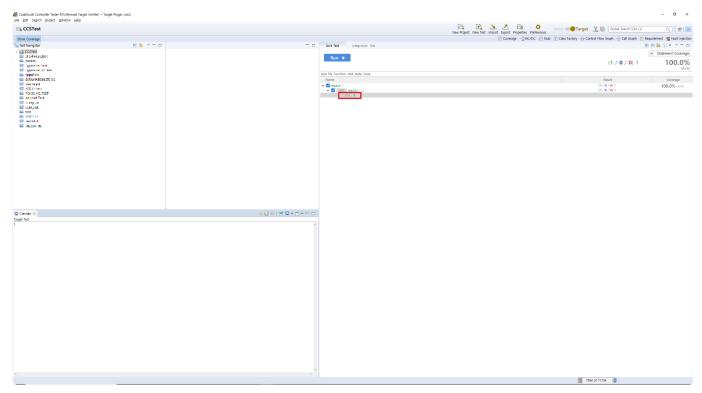
🗟 CCSTest



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

International In		Unit Test Integration Test Run	tew Riget two Tel Import Goot Ropertu ∑∶Conrege}k	AC/DC 🔃 Stub 👔 Class Factory 💠 Control Flow Graph 👘 Call Graph 💼 R	Requirement Y Fauk injectio
	- D	Run type file, function, test, state, issue Name		(0 / 0 / 0) 0 Result	Statement Coverage Coverage
Cost Cost Cost <td></td> <td>type file, function, test, state, issue Name</td> <td></td> <td>Result</td> <td>Coverage</td>		type file, function, test, state, issue Name		Result	Coverage
Cost Cost Cost <td></td> <td>type file, function, test, state, issue Name</td> <td></td> <td>Result</td> <td></td>		type file, function, test, state, issue Name		Result	
Goldenize, Suit Goldenize, Suit Goldenize, Suit Goldenize, Suit Goldenize, Suit Goldenize, Suit FOCO, Suit F		Name		Result	
Graphet Graphet Graphet PACCO_TIT PACCO_TIT Branch		Name			
Gefablobanchis Gefablobanchis monotest monotest process(stat) process(stat) process(stat) doss(stat) dos					
PACOLYNET		> C main()		(0 / 0 / 0) 0	
cooped_Tet forg_set tring_set tring_set sat sat sat sat sat sat sat sat sat					
thing_text tot t					
is study text tot is text is text1111 is text1111					
C test1111					
🔯 testtiittt					
W vdqpifet					
	import a target				
	📓 Import a targe	et test log		×	
	$\leftarrow \rightarrow + \wedge$	CT_Workspace > Workspace_3.5 > CCSTest > .csdata > ta	rget⇒ scan v č	م	
	1 . x + - 41		. III • 💷		
	💻 tu ec	^ 이름	수정한 날짜 유형	37	
	181 CC 😭	scan_result.csv			
	+ 92**	targetResult.lo	g 2021-07-15 요전 11:43 텍스트 문서	_	
Console 23	Ik 🖬 Iki 🖬 🖿 🗮 Estat				
Polarion					
	·····································				
	1 HH				
	4 74 (
	2440 10				
	_ Kannina	28.21			
	▲ HENA				
	* 1= iii	v c			
		파일 이 물란타 targetResult.log		~	
			110 HA		
			M 71(0) M 64		
< Z CCSText					

- Import the file created in step 9.



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

2. Debugger User Guides

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

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

2.1. Lauterbach TRACE32

Controller Tester can target test using the TRACE32 debugger.

Controller Tester uses TRACE32's cmm script to run tests in the target environment and get the results. A list of targets supported by TRACE32 can be found on the <u>Lauterbach homepage</u>.

- Supported target list that can generate cmm script automatically
- <u>Step1: Setting target environment in Controller Tester</u>
- Step2: Run the target test

2.1.1. Supported target list that can generate cmm script automatically

Controller Tester automatically generates a cmm script file or receives it from the user. If the cmm script can be generated automatically, you only need to enter the chip name of the target. If you cannot generate cmm scripts automatically, you must enter the cmm script file path manually. The targets that currently support the automatic generation of cmm scripts are:

PowerPC	mpc5554, mpc5553, mpc5534, mpc556x, mpc551x, mpc560xe, spc560bxx, spc560pxx, spc560sxx, mpc560xb, mpc560xp, mpc560xs, spc563m54, mpc5632m, spc563m60, mpc5633m, spc563m64, mpc5634m, mpc564xs, mpc5668, mpc5674, mpc5644a, spc564a80, mpc5642a, spc564a70, mpc567xk, spc56hk, mpc5643l, spc56el60, spc56el70, mpc5644b, mpc5644c, spc564b64, spc56ec64, mpc5645b, spc564b70, mpc5645c, spc56ec70, mpc5646b, spc564b74, mpc5646c, spc56ec74, mpc5676r, spc56ap, mpc5746m, mpc5774k, spc574k74, mpc5777m, spc57hm90, mpc574xp, mpc574xg, mpc574xr, mpc577xk, mpc577rc, 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

2.1.2. Step1: Setting target environment in Controller Tester

Select Debugger on the target environment setting page of the Controller Tester. Only a list of debuggers supported is displayed, depending on the toolchain selected for the project. Set the debugger to TRACE32.

▶ Freescale ▶ CodeWarrior-MPC55xx ▶ 2.6 ▶ others ▶ trace32

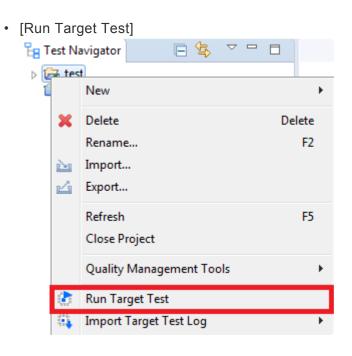
The setting items are displayed according to the selected information. The items you need to set when using the TRACE32 debugger are shown in the table below. Some of the settings are required.

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

2.1.3. Step2: Run the target test

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

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



• [Run]

Unit Test	Integration Test
Run 🕨	

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

2.1.4. Debug the target test

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

2.2. PLS Universal Debug Engine (UDE)

Controller Tester can target test using the UDE debugger.

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

A list of targets available for connection to UDE can be found on the PLS homepage.

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

- Step1: Create a workspace in UDE IDE
- <u>Step2: Setting target environment in Controller Tester</u>
- Step3: Run the target test

2.2.1. Step1: Create a workspace in UDE IDE

UDE can generate UDE workspaces from the UDE desktop IDE.

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

Select Target Configuration	
Last Used Browse	
Folder to browse :	
{UDE install path}₩Targets₩	✓ …
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 Edit Debug Show V</u> iews	<u>T</u> ools <u>C</u> onfig	W		
D New Workspace CTRL+N				
<i>🛎</i> <u>O</u> pen Workspace	CTRL+O			
💐 <u>S</u> ave Workspace	CTRL+S			
🔁 Save Workspace <u>A</u> s				
Save Workspace as <u>T</u> emplate	<u></u>			
Save View Content As	CTRL+ALT+S			
🖆 <u>C</u> lose Workspace	ALT+F4			
🖳 Load Program				
Load Program				
<u>६ C</u> onnect Target System				
Sconnect Target System Sconnect Target System	CTRL+P			
 Sconnect Target System ∑ Disconnect Target System Print Setup 	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.

PFLAS	H: 512 kByte OnChip) FLASH		•	🔽 Enable	Exit
Index	Start	End	Size	^	Erase	About
0	0x00000000	0x00007FFF	32K			
	0x00000000	0x0000007			Program	Help
	0×00000020	0x000002FF				
	0×00001000	0x000011A7			Verify	General .
	0×00002000	0x00007FFF				
1	0x00008000	0x0000BFFF	16K		HW Protect	
_	0x00008000	0x0000BFFF				
2	0x0000C000	0x0000FFFF	16K		SW Protect	
2	0x0000C000	0x0000FFFF	2014			
3	0x00010000 0x00010000	0x00017FFF	32K		Test Empty	
	0x00010000 0x00014000	0x00013F7B 0x000170D7				
4	0x00014000	0x000170D7	32K	~	Info	Program A
Rem	ove All 🕴 Rema	ove Sel.			Setup	Verify Al

See the manual provided by UDE for details.

2.2.2. Step2: Setting target environment in Controller Tester

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

▶ Freescale ▶ CodeWarrior-MPC55xx ▶ 2.6 ▶ others ▶ ude

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

Some of the settings are required.

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

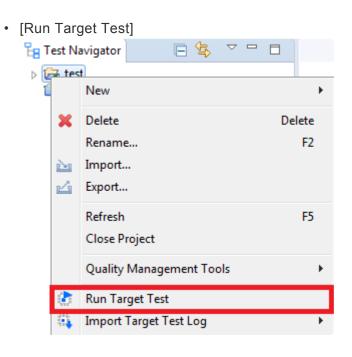
The default scripting language used by Controller Tester is visual basic script.

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

2.2.3. Step3: Run the target test

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

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



• [Run]

Unit Test	Integration Test
Run 🕨	

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

2.2.4. Debug the target test

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

2.3. iSYSTEM winIDEA Debugger

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

The list of targets supported by winIDEA can be found on the iSYSTEM home page.

The execution of the debugging script requires the python SDK installed together when installing winIDEA. If it is not installed, you can download it from the <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 Controller Tester. The iSYSTEM BlueBox iC5000 Unit debugger and NXP's MPC56xx target are used for the examples.

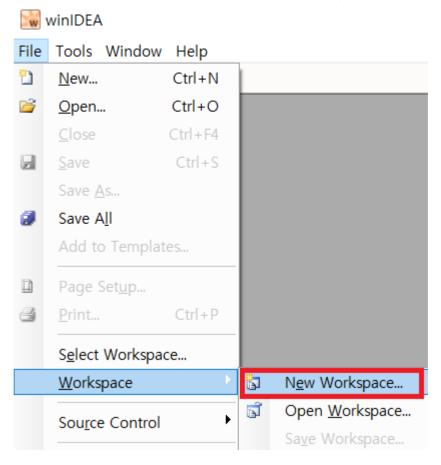
- Preparation for use of iSYSTEM winIDEA
- <u>Step1: Creating and setting up a winIDEA workspace</u>
- <u>Step2: Setting target environment in Controller Tester</u>
- <u>Step3: Run the target test</u>

2.3.1. Preparation for use of iSYSTEM winIDEA

Target testing with winIDEA in Controller Tester requires a debugger that winIDEA supports. Before running the target test, you need to create a winIDEA workspace and connect the debugger for use to the PC with Controller Tester.

2.3.2. Step1: Creating and setting up a winIDEA workspace

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



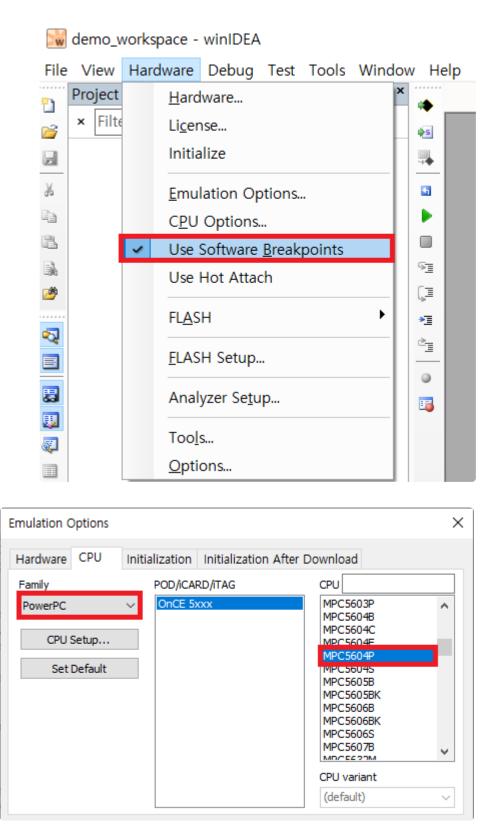
2. First, go to the top menu, select [Hardware]> [Hardware...], and then select the type of the connected BlueBox in the [Hardware Type] tab.

Hardware Configuration	×
Hardware Type Communication	
Hardware	
iC5000, iTAG2k \checkmark	
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	- Always use quality cables - Avoid hubs			
- Always use quality cables				

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	Х				
Download Files Options Target Download SMP Endian VLE Overlays	5				
Path Type Offset					
✓ MPC5604P_Example ₩bin ₩internal_FL ELF	•				
New Remove Properties Memory area default ~ Include project output file					
Default file for debugging internal_FLASH.elf	\sim				

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.

W	🔛 demo_workspace - winIDEA				
File	View Hardw	are Debug	Test	Tools Window Help	
*	<u>N</u> ew	Ctrl+N		▼ □ + □ X	
2	Open	Ctrl+O		\$ 5	
	<u>C</u> lose	Ctrl+F4			
	<u>S</u> ave	Ctrl+S		G	
	Save <u>A</u> s				
۲	Save A <u>l</u> l				
	Add to Templates			SE	
D	Page Set <u>u</u> p			Ç.	
-	Print	Ctrl+P		•≣	
	<u>_</u>	curr			
	S <u>e</u> lect Works	pace		9	
	<u>W</u> orkspace	×.	8	N <u>e</u> w Workspace	
	Sou <u>r</u> ce Contr		đ	Open <u>W</u> orkspace	
	<u>500<u>r</u>cc contra</u>			Sa <u>v</u> e Workspace	
2	<u>Find</u> In Files			Save Workspace <u>T</u> emplate	
	E <u>x</u> it	Alt+F4		Close Wor <u>k</u> space	

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

2.3.3. Step2: Setting target environment in Controller Tester

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

Set the debugger to BlueBox.

▶ Freescale ▶ CodeWarrior-MPC55xx ▶ 2.6 ▶ others ▶ bluebox

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

Required fields are displayed in red in Controller Tester.

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

The default scripting language provided by Controller Tester is python. If you use a custom debugging script, you need to write it in python to work properly. If you write in other languages, refer to the <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.

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 🕨	1

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

2.3.5. Debug the target test

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

2.4. IAR Embedded Workbench C-SPY Debugger

Controller Tester provides the ability to automatically run tests and get results in the target environment through the IAR Embedded Workbench C-SPY debugging function. The list of targets supported by C-SPY can be found on the <u>IAR website</u>.

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

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

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

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	•	φ×
Debug		~
Files	φ.	•
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	•	џ×
Debug		~
Files	φ.	•
🗆 🌒 sample_project - De	~	
⊨-⊞ 🗟 main.c		
🖵 🖅 💼 Output		

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.

Options for node "sample_p	project"	Cypress	>					
		Epson	>					
		EtaCompute	>					
Category:		Faraday	>					
General Options		Fujitsu	>					
Static Analysis Runtime Checking		HDSC	>					
C/C++ Compiler	Library Options 2 MISRA-C:2004 MISRA	Hilscher	>					
Assembler	Target Output Library Configuration Library	Holtek	>					
Output Converter Custom Build	Processor variant	Infineon	>					
Build Actions		Lapis	>	BlueNRG	>	1		
Linker		Linear	>	Other	>			
Debugger Simulator	Device None None	Maxim	>	SPEAr	>			ST STM32F429AG
CADI		Mediatek	>	STM32F0	>			ST STM32F429AI
CMSIS DAP GDB Server	O CMSIS-Pack None	Microchip	>	STM32F1	>			ST STM32F429BE
I-jet	Endian mode Floating point settings	Micronas	>	STM32F2	>			ST STM32F429BG
J-Link/J-Trace		Microsemi	>	STM32F3	>			ST STM32F429BI
TI Stellaris Nu-Link	O Big	Milandr	>	STM32F4	>	STM32F401	>	ST STM32F429IE
PE micro	BE32 D registers	MindMotion	>	STM32F7	>	STM32F405	>	ST STM32F429IG
ST-LINK	0 BE8	MitsubishiElectric	>	STM32G0	>	STM32F407	>	ST STM32F429II
Third-Party Driver TI MSP-FET		NetSilicon	>	STM32G4	>	STM32F410	>	ST STM32F429NE
TI XDS	DSP Extension	NordicSemi	>	STM32GB	>	STM32F411	>	ST STM32F429NG
	Advanced SIMD (NEON) 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	>	

 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_p	oject"	×
Options for node "sample_price 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	oject" Setup Download Images Extra Options Multicore Plugins Driver Run to I-jet Setup macros Use macro file(s) Device description file Override default	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: General Options Static Analysis Runtime Checking C/C++ Compiler	Setup Interface Trace Breakpoints	Factory Settings
Assembler Output Converter Custom Build Build Actions	Reset Connect during reset (default)	
Linker Debugger Simulator CADI CMSIS DAP GDB Server	Duration: 300 ms Delay after: 200 ms Target power Emulator Image: Comparison of the probe Always prompt for probe selection	
I-jet J-Link/J-Trace TI Stellaris Nu-Link PE micro	Leave on after debugging Switch off after debugging	
FE MICTO ST-LINK Third-Party Driver TI MSP-FET TI XDS	Log communication \$PROJ_DIR\$#cspycomm.log	
	OK Cancel	

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

2.4.3. Step3: Setting target environment in Controller Tester

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

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

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

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

Required fields are displayed in red in Controller Tester.

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

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

2.4.4. Step4: Run the target test

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

• [Run Target Test]



• [Run]



2.4.5. Debug the target test

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

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

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

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

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

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

2.5.1. Step1: Create a project in Code Composer Studio

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

😚 New CCS	S Project				\times
New CCS P		T			
Create a ne	w CCS Project.				
Target:	<select filter="" or="" text="" type=""></select>	✓ TMS320F28335			\sim
Connection:	Spectrum Digital XDS560V2 STI	M USB Emulator	~	Verify	y

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 location	
Location: C:₩ccstudio_ws₩test	Browse
Compiler version: TI v18.12.3.LTS	More
 Tool-chain Project templates and examples 	
type filter text type filter text for the selected device. for the selected device. will contain an empty 's source-file. Function of the selected device. Creates an empty project for the selected device. will contain an empty 's source-file. Creates an empty project for the selected device. Creates an empty project for the selected device. Source-file. Creates an empty project for the selected device. Creates an empty project for the selected device. Source-file. Creates an empty project for the selected device. Source-file. Creates an empty project for the selected device. Source-file. Creates an empty project for the selected device. Source-file. Creates an empty project for the selected device. Source-file. Creates an empty project for the selected device. Source-file. Creates an empty 's source-file.	The project
Open <u>Resource Explorer</u> to browse a wide selection of example projects of projects for selected devi	
Sack 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

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

2.5.2. Step2 : Setting target environment in Controller Tester

- 1. Creat a CodeScroll Controller Tester project. For more information to create the project, refer to <u>Texas Instruments 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 Controller Tester. This example selects IDE debugger to use Code Composer Studio debugger.

▶ TI ▶ C2000 ▶ 6.2 ▶ TMS320F28x I	►i	de
-----------------------------------	----	----

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.

et Properties for CCStest						
type filter text	(i) When rur	ning the targe	t test, it will build the te	st colly by filling out the required fields	on the Run tab. 🛛 🖕	• = -
Inclusion and Exclusion Info V Target test	► TI ► C2 Settings	2000 🕨 6.2 🕨	Import Target Enviro	nment Settings		
Target environment set	Analysis	Use build	script			
> Test	Build	Name		Value		
	Run	use_stdio_he		false		
	etc.	use_std_string_header		false		
		ide_director	y_path	C:\ccs1011\ccs		
		workspace		D:\CCSworkspace		
		project_name toolchain_kind		test ti		
		tooichain_ki	nu	u		
		Name:	ide_directory_path	Value: C:\ccs1011\ccs		Browse
		Description:	The path to the directory where Code Composer Studio ide is installed			
< >						
				E	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 Controller Tester

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

CT Properties for CCS								×
type filter text	When runn	ing the targe	et test, it will build ar	d run the test code.		¢	• 🔿	
	2000 🕨 6.2 🕽	Import Target Environment Settin			ettings			
Target environment set	Settings Analysis Build Run etc.	Name ccxml_path target_binal debug_prot cpu_name Name: Description:	ry_path	Value C:#Users#SURE#workspace_v9#CCSTer C:#Users#SURE#workspace_v9#CCSTer *				<u>.</u>
				E	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_probe cpu_name	Debug			
		_debug_probe	cpu_n	ame
Device Spectrum Digital XDS560V2 STM USB Emulator C28xx	Device	Spectrum Digital XDS560V2 STM	USB Emulator C28xx	

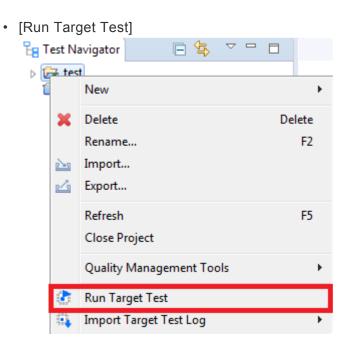
When only one debugger is connected to the target, debug_probe can be left as the

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.

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.

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

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>
- <u>Step2: Setting target environment in Controller Tester</u>
- Step3: Run the target test

2.6.1. Step1: Debugger script settings

In order to perform the target test in Controller Tester, 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 Controller Tester workspace or project name.

2.6.2. Step2: Setting target environment in Controller Tester

Select the debugger in the target test project creation wizard in Controller Tester 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 Controller Tester.

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.

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 Controller Tester, 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.

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]



3. Target Build Guide

CodeScroll Controller Tester 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

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

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

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

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

ategory:				
eneral Options tatic Analysis	•			
untime Checking				
C/C++ Compiler	Library (Options 2	MISRA-C:2004	MISRA-C:1998
Assembler	Target	Output	Library Configuration	Library Options 1
Output Converter				
Custom Build Build Actions	Library:		Description:	
Build Actions Linker	Full	~	Use the full configuration	
Debugger			library. Full locale interfac	
Simulator			descriptor support, multib and hex floats in strtod.	ytes in printr and scant,
CADI			and nex nodes in succe.	
CMSIS DAP	Configuratio	n file:		
GDB Server			DLib_Config_Full.h	
I-jet J-Link/J-Trace		interiori en	o zio_coning_i anini	
TI Stellaris	Enable th	read support	in library	
Nu-Link	-Library low	-level interfa	e implementation	CMSIS
PE micro	None		_ stdout/stderr	
ST-LINK	<u> </u>			Use CMSIS
Third-Party Driver	O Semihost		Via semihosting	DSP library
TI MSP-FET TI XDS	O IAR break	cpoint	Via SWO	
11 XDS				

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
 Target test Target environment set Test 	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	Directory path of Code Composer Studio ex.C:\ti\ccs930
workspace	Path to workspace directory in Code Composer Studio
project_name	The name of the Code Composer Studio project to be analyzed by Controller Tester

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

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

builds the target test code.

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

3.3. CodeWarrior IDE

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

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

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

Build tab

ide_directory_path	Path to CodeWarrior IDE <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'.

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'.

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'.

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

274 0 0 0 0 0 0	2 hon Jun					
8 2	CONST Master					
117, Project) (2027/07/1108_AC (Microcontroller) Pin Configurator (Design Tool)	· Being brinn the		Test/of trivel-avends and dant filologic			
Pin Carliganter Design Teel) Code Data dat (Design Teel)	- Optimization Andres antimistic		Notherief(e/)			
Code Canadar (Dealer Tool) CATHER Read Tool 5 70000 STRUMER (Debig Tool)	· Propractia		Additional include paths [7] Seture include paths [7]			
- 2 Propen Anger (Anger Tat) Sirte - 3 State - 3 State - 4 Insta	 System India's parts Record division Record division 		Nors defector(0) Nors defector(0) Nors webficitier(0)			
	v Marine		Trafformal			
	Versional and the type could be Unanitized some exactly standard library ACADs for some		No.			
	thing runded starts-rodies Libery the started iteray		old rel Nex			
	The standard FC Reary aspected Realing point data		No.			
	> they standard libraries		tang dended thereo(3)			
	> Entermine > Remorp Rocks					
	 Statut File Associative File Washing Contact Intervalue City 					
	Veriation of aution blancation file Bate Control Lot File					
	> Ofers					
	1					
	1					
	Additional include paths Specifies the additional include paths under compiling. If this option is contact, This option compared to the + option	only the standard liddens of the compiler are searched. The referen	a point of the path is the project bidles.			
	The option conseponds to the -i option.					
	Towned Sylless Temple Sylless (Scientific Sylless) and 10	and A restar to existing X distribution	er (, Tradrachonton Belother Shoe			
	Officer Officers Charles Y -					
	Runter Remap				The	tine Pro
No.fr. Picento P., ER Tenune	Californi Sana Las Californi Console Référencies (1974)	· Patients Devices Prime	Preiset Bill prove threat and the	RESIDENTIAL PROFESSION	Orr Renau PA	Dample Function
	Provide and Provid			fiscano fera	in lands for	Accord
com.codescr	per one line): 😱 oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Testi CSPlusRL78_2₩Testi	Fixture Fixture₩cs			
com.codescr	oll.ut.embedded₩Renesas	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture Fixture₩cs			
com.codescr	oll.ut.embedded₩Renesas	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture Fixture₩cs			
tcom.codescr	oll.ut.embedded₩Renesas	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
com.codescr	oll.ut.embedded₩Renesas	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
com.codescr	oll.ut.embedded₩Renesas	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
tcom.codescr tcom.codescr	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
com.codescr com.codescr com.codescr	oll.ut.embedded₩Renesas	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
com.codescr com.codescr browse Browse	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
com.codescr com.codescr browse Browse	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
tcom.codescr tcom.codescr tcom.codescr Browse Permit non- Permit non-	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Browse Permit non- Include sub	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Browse Permit non- Include sub	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Browse Permit non- Include sub laceholder:	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Browse Permit non- Include sub laceholder: Placeholder	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Browse Permit non- Include sub laceholder: Placeholder	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value	CSPlusRL78_2₩Test CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Browse Permit non- Include sub laceholder: Placeholder ActiveProject	oll.ut.embedded\#Renesas(oll.ut.embedded\#Renesas(existent path folders automatically Value Dir C:#Users\#SURE\	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Browse Permit non- Include sub laceholder: Placeholder Placeholder ActiveProjectI ActiveProjectI	oll.ut.embedded\#Renesas(oll.ut.embedded\#Renesas(existent path folders automatically Value Dir C:\Users\#SURE\ Name TEST	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Browse Permit non- Include sub laceholder: Placeholder ActiveProjectI ActiveProjectI	oll.ut.embedded\#Renesas(oll.ut.embedded\#Renesas(existent path folders automatically Value Dir C:\Users\#SURE\ Name TEST	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Browse Permit non- Include sub laceholder: Placeholder ActiveProjectI BuildModeNar	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE₩ Vame TEST ne DefaultBuild	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Com.codescr Browse Browse Permit non- Include sub laceholder: Placeholder Placeholder ActiveProjectI BuildModeNar	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE₩ Vame TEST ne DefaultBuild	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Com.codescr Browse Browse Permit non- Include sub laceholder: Iaceholder: Placeholder Placeholder ActiveProjectI BuildModeNar MainProjectDi	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE¥ Name TEST ne DefaultBuild r C:₩Users₩SURE¥	CSPlusRL78_2₩Test	Fixture₩cs			
com.codescr com.codescr browse Browse Permit non-	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE¥ Name TEST ne DefaultBuild r C:₩Users₩SURE¥	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Com.codescr Browse Browse Permit non- Include sub laceholder Include sub laceholder Placeholder Placeholder ActiveProjectI BuildModeNar MainProjectNa	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE¥ Name TEST ne DefaultBuild r C:₩Users₩SURE¥	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Com.codescr Browse Browse Permit non- Include sub laceholder: Iaceholder: Placeholder Placeholder ActiveProjectI BuildModeNar MainProjectDi	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE¥ Name TEST ne DefaultBuild r C:₩Users₩SURE¥	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Com.codescr Browse Browse Permit non- Include sub laceholder Include sub laceholder Placeholder Placeholder ActiveProjectI BuildModeNar MainProjectNa	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE¥ Name TEST ne DefaultBuild r C:₩Users₩SURE¥	CSPlusRL78_2₩Test	Fixture₩cs			
Com.codescr Com.codescr Com.codescr Com.codescr Browse Browse Permit non- Include sub laceholder Include sub laceholder Placeholder Placeholder ActiveProjectI BuildModeNar MainProjectNa	oll.ut.embedded₩Renesas(oll.ut.embedded₩Renesas(existent path folders automatically Value Dir C:₩Users₩SURE¥ Name TEST ne DefaultBuild r C:₩Users₩SURE¥	CSPlusRL78_2₩Test	Fixture₩cs			

3.7. MPLAB X IDE

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

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

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

Build tab

ide_directory_path	Installation path of MPLAB X IDE <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'.

3.8. Microsoft Visual Studio

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

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

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

Build tab

ide_directory_path	Installation path of Microsoft Visual Studio <i>ex. C:\Program Files</i> (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'.

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'.

4. Sharing Projects with Other Users

You can share the CodeScroll Controller Tester projects with others. Controller Tester 3.3 or later uses [Export Project] and [Import Project] functions.

- Guide to Share Projects
- Guide to Share RTV Projects

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

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

- Export project
- Import project

4.1.1. Export project

You can export projects, including project setup and testing.

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

File	Edit	Search	Project	Target	Window
	New				>
	Close			C	Ctrl+W
	Close All			Ctrl+Sł	nift+W
	Save				Ctrl+S
Q.	Save /	All		Ctrl+S	hift+S
	Renan	ne			F2
	Refres	h			F5
	Switch	h 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	
 > > Coverage > General > Export Project > > Metrics > > Preferences > > > Test 	
< <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.

Export Project	_		×
Export Project			
Export the project, including project settings and tests.			
Project:			
Study_sample		Select Deselec	
1 of 1 selected. Options Include Toolchain Include Source File			
Path: C:#Users#hikim#Desktop		Sear	ch
< Back Next > Finish		Cance	el l

- (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.

4.1.2. Import project

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

Import general C/C++ Project

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

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		-	– 🗆 X
Import a Project	d project.		
Project directory:	C:#Users#hikim#Downloads#test_vs2013_tch	_src_export_imported2_20220428102501	Search
Project name:	test_vs2013_tch_src_export_imported		
Location:	C:#Users#hikim#Desktop#ct_workspace		
Select Toolchain			
Default	Toolchain	Description	
	hyeintoolchain targetToolchain GCC 4.7 (32bit) GCC 5.3 (32bit)	자동으로 생성되었습니다. 자동으로 생성되었습니다.	
Options Include Toolcha			Toolchain Setting
	< B	ack Next > Finish	Cancel

• (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 the CodeScroll Controller Tester document.

- 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	
(1) 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			
(i) When ru		te the source code with the test code. You can r required fields on the Build/Run tab.	un the target test
► GNU Co	mpilers 🕨 gcc 🕨 5.3 🕨 others	nodebugger	Import Target Environment Settings
Settings			
Analysis	Name	Value	
Build	language	c	
Run	Toolchain Name	GCC 5.3 (32bit)	
	Status	This toolchain is supported.	
etc.	C Compiler	C:\gcc\5.3.0\32bit\bin\gcc.exe	
	System Header(C Compiler)	c:\gcc\5.3.0\32bit\bin\/lib/gcc/mingw3	32/5.3.0/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/mingw3	32/5.3.0/include/c++;c:\gcc\5.3.0
	Library(C++ Compiler)		
		i	
	Name:		
	Description:		
		< Back Next >	Finish Cancel

- Even if it is not a target C/C++ project, if it is a project that includes target environment settings, the target environment setting window appears when [Import Project] is executed.
 - Even if it contains an invalid path, you can complete the target environment setup and proceed to the next one, but the one-click target test may not be executed.
- 4. Complete the target environment settings and click the [Next] button.
- 5. You can check the source path included in the project to be imported. Invalid paths are marked in red and can be modified by clicking on the path list.
- 6. Click the [Finish] button.

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

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

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

- Project sharing scenario
- <u>RTV server user guide</u>

4.2.1. Project sharing scenario

When using the [Existing Projects into Workspace] function

- 1. When you create a RTV project, a RTV project directory (hereinafter referred to as RTV_A project) is created under the Controller Tester workspace.
- 2. The user who wants to share the project receives the RTV_A project directory created in the above step, and copies and pastes the RTV_A project directory into the Controller Tester workspace path that he uses.
- 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 Controller Tester test navigator view.
- 6. Right-click the RTV_A' project in [Test Navigator View] and perform [Reanalysis].
- 7. This should be done when connected to the same RTV server.

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

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

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

- 1 -startup
- 2 plugins/org.eclipse.equinox.launcher_1.4.0.v20161219-1356.jar 3 --launcher.library
- 4 plugins/org.eclipse.equinox.launcher.win32.win32.x86 64 1.1.500.v20170531-1133 5 -data
- 6 @noDefault
- -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.

port tests		
port tests xport project's test informations.		
Export path: \$(temp_folder)	Search	
Unit Test		
⊡ Test		
☑ Test Data		
Integration Test		
⊡ Test		
✓ Test Data		
Stub		
Connected Stub		
● All Stub		
Ontion		
Option Overwrite existing test files without warning		
Export only checked tests in Unit/Integration Test View.		
< Back Next > Finish	Cancel	
		-
Import tests —	X	
Import tests —	X	-
	X	
aport tests		
aport tests	Search	
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i>		
mport tests mport project's test informations.		
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 ☑ Test Data		
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test		
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☐ Test ☐ Test Integration Test Integration Test		
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test ☑ Test Data		
mport tests mport project's test informations. mport path: \$(temp_folder) Unit Test ☐ Test ☐ Test Data Integration Test ☐ Test		
mport tests mport project's test informations. mport path: <i>\$(temp_folder)</i> Unit Test ☑ Test ☑ Test Data Integration Test ☑ Test ☑ Test Data Stub		
mport 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		
mport 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		
mport 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 Integration Test Integration Test Itest Data Integration Test Itest Data Stub Connected Stub Itele All Stub Option Overwrite existing test files without warning If the same stub exists already, it is not imported.		
mport 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 Integration Test Integration Test Itest Data Integration Test Itest Data Stub Connected Stub Itele 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 Integration Test Integration Test Itest Data Integration Test Itest Data Stub Connected Stub Itele 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 Integration Test Integration Test Itest Data Integration Test Itest Data Stub Connected Stub Itele All Stub Option Overwrite existing test files without warning If the same stub exists already, it is not imported.		

4.2.2. RTV server user guide

When using one RTV server

- 1. When RTV server has a project built using the csbuild capture function
 - a. Projects can be imported according to the project sharing scenario above, without the need for additional settings.
- 2. When the RTV server is connected, but the server (IP/Port) information is different
 - a. Since server (IP/Port) information at the time of project creation is imported, existing server information is imported and toolchain information is not imported.
 - b. After modifying the server information to access the existing server, import the toolchain with the same name by importing the toolchain. At this time, the path of the tool chain used in the project should be the same.
- Sharing of RTV projects can be difficult if you are using more than one RTV server (same source file, tool chain, or if you want to receive and use a virtual machine file with RTV server installed).

5. Identifying the Cause of a Test Error

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

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

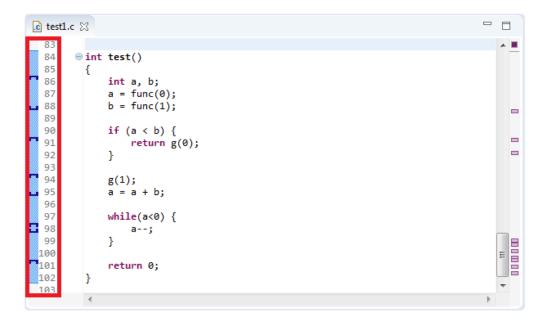
In some cases, after performing a test in Controller Tester, error results such as Signaled and Abnormal Exit are displayed. When you execute the [Inspect Debug Info] of the failed test case, the function call stack trace is displayed in the [Debug Information View]. If you added a variable/expression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

🍃 Debug Information 🔀			(×)= 🗖
Fest Case (test/func_test0) #1			
stack trace:	List of variable/expression:		
✓ ♣ [Project: test] test run	Variable/Expression	Value	Location
func3(signed int *) at test1.c:42 func2() at test1.c:30	p (test1.c: 40)	12319812	before line
func2() at test 1.c:30 func(signed int) at test1.c:12			

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.

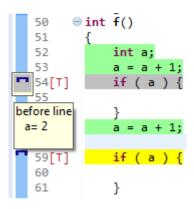
The List of variable/expression represents the variable/expression values executed with the test case.

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 [Add Variables/ Expressions to Debug] in the CodeScroll Controller Tester document.

6. Source Code Modification and Test Reconfiguration

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

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

Controller Tester divide source code modifications into four cases.

- Modifying names of test or stub functions.
- Modifying names or type of global variables used in tests.
- 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 by</u> <u>Controller Tester</u> and in cases of undetectable modification by Controller Tester, refer to <u>In Cases of</u> <u>Undetected Modification by Controller Tester</u>.

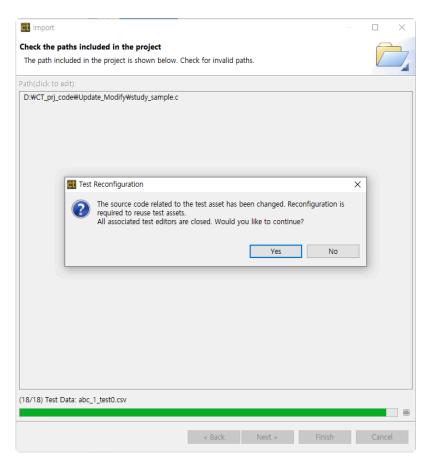
6.1. Run [Test Reconfiguration]

How to automatically use [Test Reconfiguration] feature

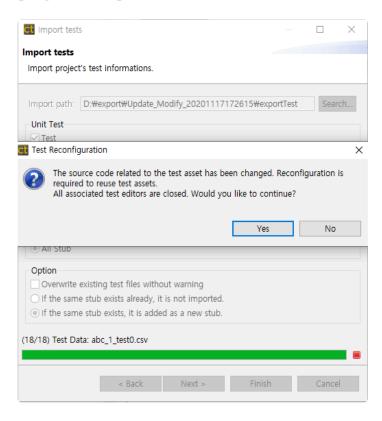
Controller Tester automatically detects following modifications.

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



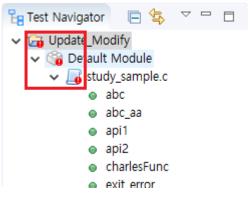
When differ present project information from imported test information using [Import test] feature



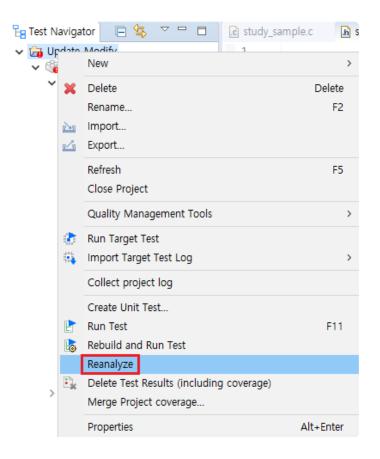
When detect source code modification after analyzing project

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

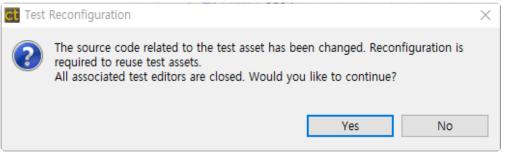
1. When the source codes modify, the Test Navigator View indicates whether the change was made.



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.

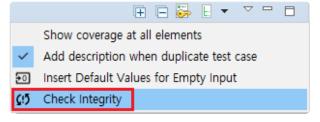
🗹 💿 main()	type filter text	I lines are displayed in red.	
	ype markex 13: 14: while (b < 5) { 15: 16: } 17: 18: a = 19: b + 5; 20: 21: for (int i = 0; i < 10; i++)	<pre>ypermetext ypermetext y= ymermetext y=</pre>	
	Inserted Before:	<pre></pre>	
	(Failed) inserted After: 1 int testVar = 0;	Inserted After:	

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

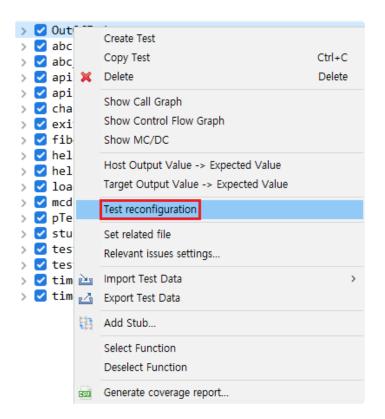
How to manually use [Test Reconfiguration] feature

If you click [No] button in [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:	64	Reconfiguring Fault Injection	
			8:	<u> </u>	Reconfiguring Fault Injection	
			9:			
			10:			
			11:	in	t a= 0;	
			12:	in	t b = 0;	
			13:			
		\checkmark	<u>14</u> :	wh	ile (b < 5) {	
		\checkmark	<u>15</u> :		a++;	
			16:	}		
			17:			
		\checkmark	<u>18</u> :	а	=	
			19:	b	+ 5;	
			20:			
			21:	fo	r (int i = 0; i < 10; i+	F) 🗸 🗸

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

6.2. In Cases of Detected Modification Automatically

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

- Modifying names of test or stub functions.
- Modifying names or type of global variables used in tests.
- 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 test or stub functions

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

abc_10			Ŧ
<pre> fibonacci_fail() ◆ fibonacci_failure() mcdcError() ◆ mcdc_Error() testMe2() ◆ testMe1() timeOutFunc() </pre>	1	Function name abc_aa0 abc0 api10 api20 main0 pTest0 helper0 testMe10 timeout0 loadFile0 mcdc_Error0 stuFunc00 exit_error0 testMeSecond0 time_out_func0 fibonacci_failure0 OutOfindex_error0	File path D:#CT_prj_code#Update_Modify#study_sample.c D:#CT_prj_code#Update

- 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.

Global variable reconfiguration	— D X
Enter the changed global variable name	
Global variables that have been renamed have been found. Please enter the changed global varial $({\ensuremath{\mathbb X}}$ Please uncheck the deleted global variables.)	le name.
Missing global variable	Enter current global variable
global0	globalO
Global variable reconfiguration	X
Enter the changed global variable name	
Global variables that have been renamed have been found. Please enter the changed global variable ($\not\!$	le 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.

Ente	er the changed global variable name	
Glo (X	bal variables that have been renamed have been found. Please enter the changed global variable Please uncheck the deleted global variables.)	r name.
	Missing global variable	Enter current global variable
\checkmark	Var_StructA[0].a	Var_StructA_New[0].a
	Var_StructA[0].b	Var_StructA_New[0].b
\checkmark	Var_StructA[0].c	Var_StructA_New[0].c
\checkmark	Var_TestA	Var_TestA_New
\checkmark	Var_TestB	Var_TestB_New
_		
	Change all variables with the same name.	
		OK Cancel

 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 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.

configure test information lefer to the previous(left) test information a	and set the new(right) test information. If you drag a variable from the old test and o	drop it on the ne	w test, the variabl	e is linked and the test data is automatically copied. [Connected: Green, Not Co	onnected: Red]	
Variables with the same name are auton Image: api1(signed int, signed int, f Image: api1(signed int, signed int, signed int, f Image: api1(signed int, signed int, signed int, signed int, f Image: api1(signed int, signed int, signed int, signed int, f Image: api1(signed int, signed	mcdc_Error(signed int, signed			mcdc_Error(signed int, signed		
 DutOfIndex_error(signed int)_0 						
mcdc_Error(signed int, signed	Test Info (Update_Modify/mcdcError_test0)		🛃 🖄	Test Info (Update_Modify/mcdc_Error_test0)	4	· 🗠
	Test Structure		Q 🕀 🖻	Test Structure	् 🖽 ।	Ξ
	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 Text global code User code Global Virable ✓ Text target function ✓ 0 mode, Emort(signed int, signed int) Local Static Variable ✓ Parameter/return Ø a : int Ø b : int Ø c : int Ø d : int	In V V V	Out ^	Name Tett global code User code Global Varable ✓ Tett taget function Local State Varable ✓ Parameter/return ● modc_prov_: int ● ● Define Before call code After call code	in Out	
	Before call code		~	User code		~
	Test Info Edit			Test Info Edit		
	Set the number format edit the partition.		^	Manage the stub items added to test.		
	Base Decimal Hexadecimal			Name Description	Add New Stub.)
	Variable Partition				Add Stub	
	~	_	Add		Remove Stub	<u> </u>
1		_	Add 2			3
	Test Info Test Case Test Code Configuration			Test Info Test Case Test Code Configuration		-

- 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.

pi1(signed int, signed int, f					api1(signed int, char, float,		
est Info (Update_Modify/api1_test0) 🛃 🖄		i 🔤		Test Info (Update_Modify/api1_test1)		42	
Test Structure		् 🖬	-		Test Structure		1725 IT
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.	9	. 🏵 E
Name	In	Out	^		Name	In	Out ^
 Test target function 					 Test target function 		
 api1(signed int, signed int, float, unsigned long, long double) 					 api1(signed int, char, float, unsigned long, long double) 		
Local Static Variable					Local Static Variable		
 Parameter/return 					 Parameter/return 		
○ a : int					o api1_a : int o a : int		
● b : int	×			T.	o b : char		
 d : unsigned long 					 d : unsigned long 		
e e : long double	~				e e : long double		

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				
Reconfiguring Fault Injection				
Refer to the old fault injection information((X Functions with fewer fault injection enable)	left) and configure the new fault injection information(right). Changed oled than before are displayed in orange)	lines are displayed	d in red.	
✓	type filter text		type filter text	
	<pre>13: 14: while (b < 5) { 15: a++; 16: } 17: 18: a = 19: b + 5; 20:</pre>	^	13: ✓ 14: while (b < 5) { ✓ 15: a++; 16: } 17: 18: a = 19: b + 5; 20:	^
	<pre>21: for (int i = 0; i < 10; i++) 22: {</pre>	~	<pre>21: for (int i = 0; i < 10; i++) 22: {</pre>	~
	inserted Before: 1 /* test */	^	Inserted Before: 1 /* test */	^
	<	>	- <	>
	(Failed) Inserted After: 1 int testVar = 0;	^	Inserted After: 1	^
1	<	?	¢	,3
			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.

6.3. In Cases of Undetected Modification Automatically

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

- Modify value type of global variable 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.

7. Navigate Source Codes

Controller Tester 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 >	P	Project Explorer
	Cut Copy Paste	Ctrl+X Ctrl+C Ctrl+V		Outline Navigator System Explorer Properties
	Source	Alt+Shift+S >		
	Search Text	>		
(x)= (전)	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.

ジ 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+
Create Stub		Project	
Add variable/expression to debug		File	
View a list of variable/expression to deb	ug	Working Set	
Preferences			

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

8. 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

8.1. 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)
}
```

8.2. 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.

8.3. 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

8.4. 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

8.4.1. 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 ×					- E
Test Info (poco/CppUnit_TestDecorator_TestDecorator_t	test0)				<u> 2</u> 2
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	
 Test target function 				Use class code	
 				Use mock	
Local Static Variable				Function for object return.	
 Parameter/return 					
✓ • test : class CppUnit::Test *					
🗸 🍳 🍾 test_mem : class CppUnit::Test					
M CS_MOCK_CppUnit_Test					
Before call code					
After call code			×		

8.4.2. 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					

- 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	Generation mock specification						
	t and return value and number of function calls for the se rou click OK, the mock specification is automatically gener						
- 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					
	ОК	Cancel					

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

Type	Input value
	any value
	any value
	<user input=""></user>
	Type class CppUnit::TestResult *

- 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	
i Sets strin	the order of return values when function called repeatedly. When entering a value, charac g type values must be enclosed in '' (quotation marks) and "" (double quotation marks).	ter 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.

ration mock specification	\times						
Generation mock specification							
the number of function repetitions to set.							
of repetitions	-						
No number of repetitions specified 🗸							
No function call							
OK Cancel							
	on mock specification the number of function repetitions to set. of repetitions No number of repetitions specified v No function call						

- 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.

9. 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		← ▼ ⇒ ▼ ▼
type filter text Analysis Editor Exclusion Language Performance Source File Types Target Test Coverage Export test data External editor Perspective Type Partition Unit Test View Virtual Address Theme (CS Labs)	Configuration n		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	eate	×
Virtual mem	ory address Co	nfiguration	
 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>)
		ОК	Cancel

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

<u>2</u>2

pe filter text	Virtual Address		<p -="" th="" ⇒<=""></p>
Inclusion and Exclusion Info Target test Test	Configuration name: REG_BASE		~
Exclusion of coverage External Command Senerate test cases aut Run Test Virtual Address	Begin 0xFFE40000	End 0xFFE40100	
			^
			~
>			

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	0	. 🗉 🖻	Test Info Edit
Test structure using a tree view and edit the information in the test.			Edit code to be executed before the function call
Name Test global code User code Global Variable	In	Out	<pre>1 { 2 int base_addr_value = CS_INT_INPUT(int, "base_addr_value"); 3 CS_ADDR_SET(int,0xFFE40000,base_addr_value); 4 }</pre>
Test target function Jest target function Local Static Variable			
 ✓ Parameter/return o^R returnVar : int ✓ Before call code 		V	
berde call code base_addr_value : int After call code	V		
User code V Stub			
Host Target			<
Test Info Test Case Test Code Configuration	_		

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

5. Edit test case values

	st Case (Virtual memory example/testMe_test0) #1							
arar	meter	Туре	Input	Expected Value	Host Output	Target Output		
	Test global code							
	User code							
	Global Variable							
/	Test target function							
~	 							
	Local Static Variable							
	> Parameter/return							
	 Before call code 							
	 base_addr_value 	int	10					
	After call code							
	User code							
>	Stub							

Test Info Test Case Test Code Configuration

10. Guides to Import Coverages

When importing coverages from Controller Tester 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>

10.1. Import Coverages by Version

When Controller Tester import coverages, it checks the version of the coverage shared file.

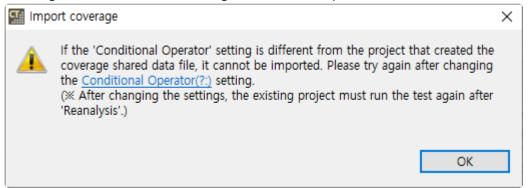
• When importing the higher-versioned coverage shared file, the coverage cannot be imported.

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.

10.2. Import Coverages by Conditional Operation Option

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



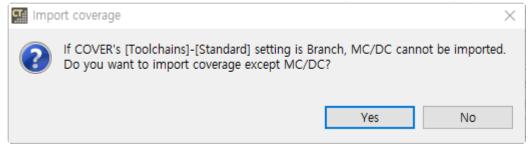
Click the link in the warning window or [Preferences] > [Test] > [Coverage] > Branch coverage, MC/DC measurement operator]. Then, change the [Conditional Operator(?:)] option to match the file you want to import.

- 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].

10.3. 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.