CODESCROLL
CONTROLLER TESTER

Controller Tester Troubleshooting Guides

3.6 — Last update: Nov 30, 2021

Suresofttech

Table of Contents

1. Controller Tester Troubleshooting Guide 3
1.1. How to assign values to flexible array members 5
1.2. How to increase heap memory in JAVA when memory is low 6
1.3. Project DB file (*.csp) is damaged due to abnormal termination 7
1.4. Testrun.exe crashes when testing QNX software 8
1.5. When a specific function is not displayed in the coverage view 9
1.6. If the error message is not displayed after the project analysis fails 10
1.7. The screen is cut off when the Windows display magnification is not 100% 11
1.8. Windows.h file not found 12
1.9. If INFO [ut.hio]: runTest:testrun exit code(105) is displayed in the log and the unit test is not executed 13
1.10. When “Extract Toolchain Info” fails 14
1.11. Messages display abnormally in the error view 15
1.12. When testing with source code containing C++20 items 16
1.13. Errors that may occur during test execution 17
1.14. When cannot find global values after importing integration test 18
1.15. “C2118 : negative subscript” error occurs during test execution 19
1.16. SDK version issue when using Visual Studio 2015 toolchain 20
1.17. Issues during installation of Controller Tester 21
1.18. When cannot save a large control flow graph as a image file 23
1.19. When workspace is broken 24
1.20. Error occurs when upload data to VPES 25
1.21. Metric view error 26
1.22. How to check values of local variables and apply to results of test cases 27
1.23. When results of function call coverage and function coverage are not consistent 28
1.24. Issue about expected and output values of floating variables 29
1.25. When fail to import coverages of COVER 31
1.26. Signal error occurs when using a constant address 32
1.27. When comparing a variable of a floating point type, the result is different. 33
1.28. When running tests, test execution fails after build is successful 35
1.29. When only “0” exists in the partition list of the stub variable. 36
1.30. When stub input data information is deleted 37
1.31. When a signal error occurs in a function that takes a variable array as an argument 38
1.32. When the size of the structure is different from Controller Tester and in the original program 39
1.33. When an undefined referenced error occurs in the IAR environment 40
1.34. When an error occurs in the test code due to the existence of a const global variable 42
1.35. When a signal error occurs when accessing a static const pointer variable 44
1.36. “C2512: No suitable default constructor available.” If an error occurs 45
1.37. How to design stub code that behaves differently for each test case 46
1.38. When perspective of Controller Tester is broken 47
1.39. Problem that Korean comments are broken when exporting stub code before Controller Tester 3.4 and importing after 3.4 48
1.40. When a value cannot be entered in the test case because the register variable is declared as a

macro ... 49
1.41. How to convert a report in xls format to html format ... 50
1.42. “LoadLibrary failed with error 87: The parameter is incorrect.” error occurs ... 51
1.43. Unmanaged code variable list cannot be displayed in the test editor (for versions before CT 3.4) ... 52
1.44. How to enter invalid values in test data ... 53
1.45. How to enter a NULL value for a pointer-type parameter ... 55
1.46. When using the conversion toolchain, the coverages are displayed abnormally in the integration test with the infinite loop removed. ... 57
1.47. If the file name is truncated when exporting tests ... 59
1.48. When an error ‘invalid use of void expression’ occurs in the host test using the conversion toolchain ... 60
1.49. When tests need to be created only for new functions in a recycling scenario ... 61
1.50. If the color of the covered area does not change when selecting a test case ... 64
1.51. Problem that the project does not open when changing the version from CT2.9 to CT3.0 ... 65
1.52. How to set only one test case to be created when creating a test ... 66
1.53. Integrated coverage is displayed as 0 or the execution result is not visible after the test run ... 67
1.54. (After Ver.3.6) When you want to import test data as ‘overwrite’ ... 68

2. Controller Tester Target Plug-in Troubleshooting Guide ... 69
2.1. Build issues after exporting target test code ... 70
2.1.1. When the entry point function name is not main (for versions before CT 3.2) ... 71
2.1.2. Multiple definition error of the function used to save the target test result (for versions before CT 3.2) ... 72
2.1.3. ‘sprintf’ has not been declared or CS_FLT_OUTPUT error (for versions before CT 3.2) ... 73
2.1.4. Target log interface settings ... 74
2.1.5. Check for signal errors ... 75
2.1.6. Problems When Testing Functions Related to UART Communication in Target Software ... 76
2.1.7. When building with CodeWarrior, test related files such as cs_tfx.c are not found ... 77
2.1.8. Target output value does not appear or is output as 0 for float type in CodeWarrior ... 78
2.1.9. undefined error of type codescroll_int32 and codescroll_uint32 ... 79
2.1.10. When cannot use ‘long’ or undefined type to ‘long’ error occurs in codescroll_int, codescroll_uint type ... 80
2.1.11. When the address of cs_io_putbyte is not found in Code Composer Studio ... 81
2.1.12. When declaration is incompatible with “void cs_io_putbyte” error occurs in GreenHills AdaMulti ... 82
2.2. TRACE32 related issues ... 83
2.2.1. symbol not found error “ct_target_log” ... 84
2.2.2. target reset failed ... 85
2.2.3. ‘&binary_path’ symbol not found or an error occurred at Data.Load.Elf “{file_path}” /LPATH while running the cmm script ... 86
2.3. Importing target log (test results) issues ... 87
2.3.1. Failed or Error When Performing ‘Import Target Test Log’ ... 88
2.3.2. When the target log buffer size is exceeded ... 89
2.3.3. When an exception occurs in the script when running after test building in Code Composer Studio ... 90

2.3.4. Fail to import debug information logs in target test 91
2.3.5. How to read target logs when fail ‘Import Target Test Log’ 92
2.4. Other tips 93
2.4.1. .map file 94
2.4.2. TRACE32 debugging 95
2.4.3. If the output value is different due to the byte order of the host/target environment 96

1. Controller Tester Troubleshooting Guide

This is a guide to solving common problems you may encounter using Controller Tester.

- How to assign values to flexible array members
- How to increase heap memory in JAVA when memory is low
- Project DB file (*.csp) is damaged due to abnormal termination
- Testrun.exe crashes when testing QNX software
- When a specific function is not displayed in the coverage view
- If the error message is not displayed after the project analysis fails
- The screen is cut off when the Windows display magnification is not 100%
- Windows.h file not found
- If INFO [ut.hio]: runTest:testrun exit code(105) is displayed in the log and the unit test is not executed
- When “Extract Toolchain Info” fails
- Messages display abnormally in the error view
- When testing with source code containing C++20 items
- Errors that may occur during test execution
- When cannot find global values after importing integration test
- “C2118 : negative subscript” error occurs during test execution
- SDK version issue when using Visual Studio 2015 toolchain
- Issues during installation of Controller Tester
- When cannot save a large control flow graph as a image file
- When workspace is broken
- Error occurs when upload data to VPES
- Metric view error
- How to check values of local variables and apply to results of test cases
- When results of function call coverage and function coverage are not consistent
- Issue about expected and output values of floating variables
- When fail to import coverages of COVER
- Signal error occurs when using a constant address
- When comparing a variable of a floating point type, the result is different.
- When running tests, test execution fails after build is successful
- When only “0” exists in the partition list of the stub variable
- When stub input data information is deleted
- When a signal error occurs in a function that takes a variable array as an argument
- When the size of the structure is different from Controller Tester and in the original program
- When an undefined referenced error occurs in the IAR environment
- When an error occurs in the test code due to the existence of a const global variable
- “When a signal error occurs when accessing a static const pointer variable”:”#static-const-signal-error
- “ “C2512: No suitable default constructor available.” If an error occurs”:#c2512
- How to design stub code that behaves differently for each test case
- When perspective of Controller Tester is broken
- Problem that Korean comments are broken when exporting stub code before Controller Tester 3.4 and importing after 3.4
- When a value cannot be entered in the test case because the register variable is declared as a

macro
- “How to convert a report in xls format to html format”:”#xls-html
- “ “LoadLibrary failed with error 87: The parameter is incorrect.” error occurs”:#quot-loadlibrary-failed-with-error-87-quot
- Unmanaged code variable list cannot be displayed in the test editor
- How to enter invalid values in test data
- How to enter a NULL value for a pointer-type parameter
- When using the conversion toolchain, the coverages are displayed abnormally in the integration test with the infinite loop removed.
- If the file name is truncated when exporting tests
- When an error ‘invalid use of void expression’ occurs in the host test using the conversion toolchain
- When tests need to be created only for new functions in a recycling scenario
- If the color of the covered area does not change when selecting a test case
- Problem that the project does not open when changing the version from CT2.9 to CT3.0
- How to set only one test case to be created when creating a test
- Integrated coverage is displayed as 0 or the execution result is not visible after the test run
- When you want to import test data as ‘overwrite’

1.1. How to assign values to flexible array members

Here’s how to assign values to flexible array members added in c99.

```
#include <stdio.h>

typedef struct _line {
        int size;
        int data[];
}line;

void funfun(line * abc){
        if(abc->data[0] == 1 ){
                printf("You can't touch this");

        }
        else{
                printf("MC Hammer");
        }

}
```

To design a test so that the if branch in the code above is true, it is not possible to put a value in the usual test data entry method.
This is because it is a flexible array member.

Therefore, the value should be entered as shown below using the code before the call in the test information tab.

```
size_t this_length = 5;
abc = (line *)malloc (sizeof (line) + this_length);
abc->data[0] = 1;
```

! You cannot use the above method in an environment without malloc.

1.2. How to increase heap memory in JAVA when memory is low

When the Controller Tester written in Java language based on Eclipse RCP and executed in jvm runs out of memory, the performance may deteriorate due to frequent GC (Garbage collection) or the program may not operate normally due to insufficient memory.
In this case, you can use the Controller Tester 64bit package (maximum heap memory default value of 8G), or manually adjust the heap memory settings.

Here’s how to manually adjust the heap memory settings.

1. Open the *(product installation path)* \CodeScroll.ini file with an editor(requires modification with administrator privileges).
2. Run Controller Tester again, increasing the value of -Xmx. Controller Tester cannot be executed if too large a value is entered. Find and set the maximum value by decreasing or increasing the value little by little.

✻ The maximum value of Xmx depends on the free memory status of the PC. In general, when using 4GByte memory and executing only Controller Tester, it will have a value in the range of approximately 1000MByte ~ 2000MByte.

1.3. Project DB file (*.csp) is damaged due to abnormal termination

Controller Tester saves project information in SQLite Database file.

```
(Workspace path)\(project name)\.csdata\(project name).csp
```

DB may be damaged when Controller Tester is abnormally terminated during operation due to a power failure or a Windows error.
In this case, you can recover the damaged DB file in the following way(not always possible).

1. Run command prompt from below path
   - `(Product installation path)\plugins\com.codescroll.gp.core_1.0.2.201202152119\lib`
2. Run the below command at the command prompt
   - `$ sqlite3 corrupted.db ".dump" | sqlite3 new.db`

1.4. Testrun.exe crashes when testing QNX software

When testing C++ code, if the global variable is of class type, abnormal termination may occur in the constructor of the class.

In this case, you should check the following:

1. If the __get_errno_ptr() function is made of stubs, add the following to the stub body.
   - `int* x = (int *)calloc(1, sizeof(int)); return x;`
2. If you are using a debug utility or a utility related to shm (shared memory), you need to make sure that the file path is not hardcoded. If it is hard coded, you should create system functions (fprintf, etc.) to access the file as stubs.
3. You need to make sure that you do exit code at initialization time. If you run the exit code, you must generate the exit code as a stub.
4. You need to check if the constructor stub has a throw. If there is a throw, it should be removed.

1.5. When a specific function is not displayed in the coverage view

If the function is not displayed in the coverage view and the solution is as follows:

Function not showing in coverage view	Resolution
If the source file to which the function belongs is excluded from analysis	Remove the corresponding source file or included path from [Inclusion and Exclusion] of project properties or [Exclusion] of environment setting
The function or the source file to which the function belongs is selected as an exclusion of coverage measurement	Remove the function or source file from [Exclusion of coverage] of project properties
If the function is not executed	Check whether the test was executed normally or check the design of the test case

1.6. If the error message is not displayed after the project analysis fails

When creating a unit test after project creation (analysis), if it fails without any special error message, please check the following:

Analysis fails without message	**Resolution**
Controller Tester is not run with administrator privileges	Redo as administrator
The path of the generated project or the path or name of the source file to be tested is very long.	Modify the path of the workspace or source file under test to the shortest path possible
If there is a path related to MYSQL in the PATH of the environment variable, and the path contains “&”	Restart Controller Tester after deleting MYSQL path if “&” is present

1.7. The screen is cut off when the Windows display magnification is not 100%

If you set the Windows display magnification to a value other than 100%, setting the following will solve the problem that the screen is cut off or the image is broken.
(Controller Tester automatically reflected from version 3.1)

1. Right-click the executable file (CodeScroll.exe) or shortcut file and click [Properties]
2. On the [Compatibility] tab, click Change High DPI Settings
3. In [Override High DPI Adjustment], select Override High DPI Adjustment Behavior.
4. Select [Coordinator] as the system and confirm

If you set it as above, the Windows system will automatically adjust the scale, so the screen may appear slightly whitish.

1.8. Windows.h file not found

When installing the Windows SDK 6.1 and using the Visual Studio 2008 compiler, the following error message may be displayed in certain environments.

```
Cannot open include file: 'windows.h': No such file or directory
```

This error is caused by missing content in the batch file that sets up the Visual Studio build environment. The solution is as follows.

Search for and locate the windows.h header file in the C:\Program Files\Microsoft SDKs directory.
`Example) C:\Program Files\Microsoft SDKs\Windows\v6.1\Include`

Open the file C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\bin\vcvars32.bat with an editor and add the path SDKs\windows\v6.0A\include, SDKs\windows\v6.0A\lib to the INCLUDE, LIB, and LIBPATH variables.

```
Example) line 26  @set Path to add to INCLUDE: C:\Program Files\Microsoft SDKs\
Windows\v6.1\Include

@set INCLUDE=C:\Program Files\Microsoft SDKs\Windows\v6.1\Include;%VCINSTALLDI
R%\ATLMFC\INCLUDE;%VCINSTALLDIR%\INCLUDE;%INCLUDE%\
```

1.9. If INFO [ut.hio]: runTest:testrun exit code(105) is displayed in the log and the unit test is not executed

The message INFO [ut.hio]: runTest:testrun exit code(105) is displayed in (project name)_full.log, and the unit test may not be executed.

In this case,
`(Worksapce)\(project name)\.csdata\build\testrun.log`
Check the file for the following error message.
bc. error:add virtual address : memory alloc error [E000000-E1FFFFFF]
error:fail to init iohandle(105)

This error occurs when the address range set in [Virtual Address] in the preferences settings is invalid.

In the [Virtual Address] of the environment setting, modify the memory address range that can be used in the current system.

1.10. When “Extract Toolchain Info” fails

When the [Extract Toolchain Info] button is clicked in the Add a Toolchain Wizard, it may fail.
In this case, you can solve the problem by taking the following measures.

- In [Toolchain Information] of the Add a Toolchain Wizard, enter the environment information file (*.bat) to run the compiler in [Env Script].
    - Example) For the Visual Studio cl compiler, enter the vcvars32.bat file in the same path as the cl.exe file.
- Check if the security program (such as ALYac) has blocked the tce.exe executable and add it to the scan exceptions list.
- Check if any other value besides the default value is added to ComSpec variable among Windows environment variables, and if so, modify it to the default value.
    - Default value for ComSpec variable: ComSpec=C:\Windows\system32\cmd.exe

1.11. Messages display abnormally in the error view

Depending on the environment in which Controller Tester is used, the encoding of the error message that is output may be different, and the message may be displayed abnormally.
In this case, you can solve the problem in the following way.

`(Product install path)\plugins\com.codescroll.gp.rcp.helios_1.0.0.201909240351\plugin_customization.ini`
Open the file in an editor.

in line 85,

```
#log file encoding (log plugin)
com.codescroll.gp.log/log.file.encoding=euc-kr
```

After changing euc-kr to the encoding suitable for the current environment, restart Controller Tester.

1.12. When testing with source code containing C++20 items

The analyzer currently used by Controller Tester does not support C++20 items and some new headers, so if you want to test with source code that includes those items, you need to do something extra.

1. Edit *(Controller Tester global path)* \1.1\parserConfig\ *vs2019 toolchain name*.conf
   - Add “ms_c++20” to the last line
2. Edit *(Controller Tester install path)* \plugins\com.codescroll.ut_3.3.2\config\cl.flag.txt file
   - Add “&/std:c++latest” to the last line of
   - This value must be cleared again when analyzing/performing sources that do not contain new items.

1.13. Errors that may occur during test execution

1. Error that can’t use default constructors.

Example code

```
class A{
public:
        A(int a, int b){}
        testFunction();
}
```

- When generate tests about class member functions, Controller Tester calls the functions and measure the coverages after create instances with default constructors. In the example code above, there is no default constructor of `class A`, so an error may occur when creating an instance. When this error occurs, you need to modify the test to create an instance with appropriate constructor.

2. Error that can’t create an instance

- When create tests about abstract class, an error may occur by creating an instance of the abstract class and trying to call the function on the created instance. In this case, find the class in ‘Test Info” and change ‘Constructor’ into ‘User code’. After that, modify the user code to create an instance of the class that inherits the abstract class.

1.14. When cannot find global values after importing integration test

An undefined reference error may occur for a specific global variable when executing a test after importing the integration test data, which was performed well in the past. If a ‘Not Found element’ message is displayed when you click ‘Global Variable’ in the ‘Test Info’ of the test in which the error occurred, the TU of the global variable may have been changed due to code modification. If you change the related file in the ‘Configuration’ tab of the test to the source file that the TU of the global variable was modified, you can check that the test works normally.

1.15. “C2118 : negative subscript” error occurs during test execution

This error message may occur when use Visual Studio toolchain or conversion toolchain. Remove `/Zp` option added on ‘Compile flag’ of ‘Module’ – ‘Properties’.

1.16. SDK version issue when using Visual Studio 2015 toolchain

If SDK version 8.1 and 10.0.10240.0 or higher are installed together when using Visual Studio 2015 toolchain, tests may not be run normally. This is an issue because `vcvars32.bat` environment setting script of Visual Studio 2015 is set to the latest SDK version. Add system headers of the latest SDK version with `/I` option to ‘Compile flag’ in ‘Module’ > ‘Properties’.

Ex) when latest SDK version is 10.0.18362.0
/I“C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\include”
/I“C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\atlmfc\include”
/I“C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\ucrt”
/I“C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\um”
/I“C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\shared”
/I“C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\winrt”
/I“C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\cppwinrt”

1.17. Issues during installation of Controller Tester

When the installer cannot access because the installation file is in a drive, not in a normal disk, or when security programs are affect to installation

1. Move the installation file to system drive.
2. Right-click the file and execute it as administrator.

When the installation file is broken during transmission

Download the file again and re-install.

The others

When you try the above methods and the same failure occurs, try to the below.

1. open the command prompt(cmd.exe) and move to the path of installation file.
2. execute the installation wizard using the below command.
   - when install using .msi file
     ```
     msiexec /I "installation_file_name.msi" /L*V "install.log"
     ```
   - when install using .exe file
     ```
     "installation_file_name.exe" /L*v "install.log" /I
     ```
3. After executing the installation wizard, *install.log* file is made. If the following log is in the *install.log* file, it is an issue about Windows Installer.
   ```
   MainEngineThread is returning 1603
   ```
4. In this case, reboot the computer.
5. Open Task Manager and teminate *msiexec.exe* process if it is running. Then re-install Controller Tester.
6. If the problem is not solved yet, try the below troubleshooting guides.
   https://support.microsoft.com/en-us/help/17588/windows-fix-problems-that-block-programs-being-installed-or-removed
   https://support.microsoft.com/en-us/help/834484/you-receive-an-error-1603-a-fatal-error-occurred-during-installation
7. If the problem is not solve after following the Window troubleshooting guides, send the below informations to the technical support.
   - Package details that you try to install.
     - ex) Controller Tester (Host) 3.4.2 ×64
   - *install.log* file that made during installation by command prompt.
   - Whether this problem occurs in particular PC.
   - Environment of PC that this problem occurs.
   - (When you use other products of our company) When install other products of our company, the same problem occurs.

Technical support contact

- help@suresofttech.com

1.18. When cannot save a large control flow graph as a image file

When export a graph as an image file, Controller Tester temporarily stores to memory and exports it. `Not enough memory` error can occurs when the size of image is huge — the size is over 512M. In this case, you can get the graph image in three ways.

- Modify the `-Xmx` option `CodeScroll.ini` file in Controller Tester install path to the maximum of used system.
  - ex) `-Xmx1200m`
- Copy to clipboard instead of exporting as an image file and paste to other program.
  - This method allows larger sizes than exporting as images.
- After expoting as an graph format(ex. ygf), open the file in graph editor like yEd. Then copy to clipboard or export as PDF format.

1.19. When workspace is broken

When Controller Tester workspace is broken, you may solve the problem, following the below.

1. Create a new workspace and a new project with a same name as previous project. Then, close Controller Tester.
2. Using the Windows file explorer, copy all contents of previous project directory and paste to the new project directory.
3. Restart Controller Tester.

1.20. Error occurs when upload data to VPES

The below error can occur when upload data to VPES

```
There are changes in the project. Please run the project and try again.
```

In this case, reanalyze the project and try again.

1.21. Metric view error

Try the following way when an error occurs in **Metric View**.

1. Close Controller Tester.
2. Delete `C:\Users\%user_name%\AppData\Roaming\CodeScroll\1.1\metric_messages` directory.
3. Restart Controller Tester.

1.22. How to check values of local variables and apply to results of test cases

You can use fault injection(code injection) and test macros to determine the success/failure of a test case based on the value of a local variable at a specific point in the source code. This method allows limited validation of local variables.

In fault injection view, insert the following code in the point where you want to check the value of the local variable.

```
if(CS_TESTCASENO() == 1) {
        CS_ASSERT(temp == 0);
}
```

After inserting the code and running the tests, test case #1 fails if value of `temp` in the corresponding part is not 0.

1.23. When results of function call coverage and function coverage are not consistent

In Controller Tester, there are cases that function call coverage is 100% but function coverage is not 100%. This difference occurs because there are difference in measurement of function call coverage and function coverage. In the case of function coverage, Controller Tester measures whether the function was actually called during the test. Function call coverage is measured whether the code block(a node in the control flow graph) that calls the function is executed.

ex) `if ( a()==1 || b()==1 )`

If you have the above code, there are cases that function call coverage and function coverage are not consistent. If `a()==1` is *true*, skip without calling `b()`. At this time, since `a()` and `b()` are the same code block, the function call coverage is displayed as `2/2`, but since only `a()` was actually called, the function coverage is displayed as `1/2`.

1.24. Issue about expected and output values of floating variables

When the test case does not fail even if the expected value and the output value of the floating variable are different.

In Controller Tester, the tolerance of floating variables is set to the 6th place after the decimal point. For example, if the expected value is 3.14159265 and the output value is 3.14159274, Controller Tester determines that the expected value and the output value are the same, and the test case is successful.You can change the tolerance in the following way.

- Open `%project_path%\.csdata\ut.ini` file and modify the value of `FLOAT_TOLERANCE`.
    - In the example above, modify to `FLOAT_TOLERANCE=0.0000001`.

When the expected value and the output value of a floating variable are the same but the test case fails.

When Controller Tester handles *float* variables, an error can occur because it casts *float* variables to *double*. This problem can be solved in two ways.

1. Modify declaration of *float* variable to declaration of *double* variable.
2. Modify `codescroll_flt` type from *long double* to *float*.
    - Open the `.info` file of the used toolchain in ‘Preferences’ > ‘ToolChain’ > ‘Open Configuration Folder’ and modify as the below.

```
// Before modification
#typeName,valueKind,min,max,size,csType
long double,float,2.22507e-308,1.79769e+308,8,codescroll_flt

// After modification
#typeName,valueKind,min,max,size,csType
float,float,2.22507e-308,1.79769e+308,8,codescroll_flt
```

When calculating the value of a floating variable using a constant, the operation result is incorrect.

If you use constants in expressions, Controller Tester converts real numbers to integers and calculates them. When doing calculations using constants, you need to change an option so that real numbers are not converted to integers.

- Open `.ini` file of the used toolchain in ‘Preferences’ > ‘ToolChain’ > ‘Open Configuration Folder’ and modify as the below.

`leave_float_literal = 0` -> `leave_float_literal = 1`

This problem can occur when using RTV project or RTV target project.

1.25. When fail to import coverages of COVER

when the compile flag used by COVER and Controller Tester are different.

The preprocessed files are used to measure coverages. Different compile flags may result in different preporcessed files. In this case, set the compile flags of COVER and Controller Tester identically, execute the tests again, and export/import coverages.

```
void testFunction(){
    int a;
    int b;
//In the case of adding CT_FLAG macro in Controller Tester and not adding it i
n COVER, or vice versa.
#ifdef CT_FLAG
     callFunction();
#endif
}
```

When differently calculate the *bodyhash* of the same funcion.

Importing coverage may fail if the *bodyhash* value of the same function is differently calculated in COVER and Controller Tester In this case, add `EXCLUDE_BODYHASH_CVG_IMPORT=true` option to `ut.ini` file, so that you can import coverage even if the *bodyhash* values are different.

- `ut.ini` path : `project_path\.csdata\ut.ini`

1.26. Signal error occurs when using a constant address

```
int * ptr;
int * pre_ptr = (int *) (0x60000000U);
ptr = (int *)pre_ptr ; // use pointer casting
```

If there is a code that allocates a constant address value to a pointer as above, a signal error may occur when the test is executed.
At this time, if the use_memory_map=1 option is set in the .ini file of the toolchain used in the project, the signal error can be resolved by setting the virtual memory address to be used in the project.
You can use the following method to set the virtual memory address in the project.

- After adding the virtual memory address you want to use in [Preferences]> [Test]> [Virtual Address], select [Project]> [Properties]> [Test]> [Virtual Address] and set it up.

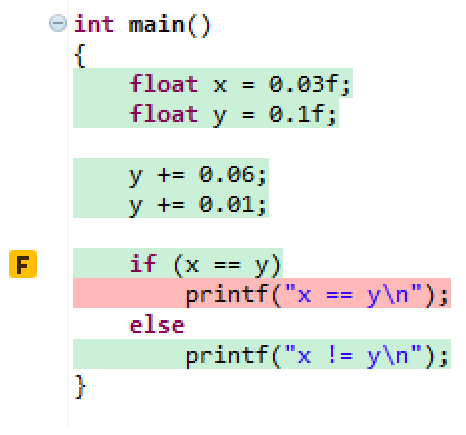
After setting the virtual memory address in the above method, reanalyze the project and run the test to confirm that no signal error occurs.

You can check the toolchain's .ini file by executing [Preferences]> [Toolchain]> [Open Configuration Folder]. When writing an option, it must be written under [CONVERTER_OPTION].

When using pointer casting for regular pointer variables, both the use_memory_map and convert_pointer_cast_variable options must be turned on 1.

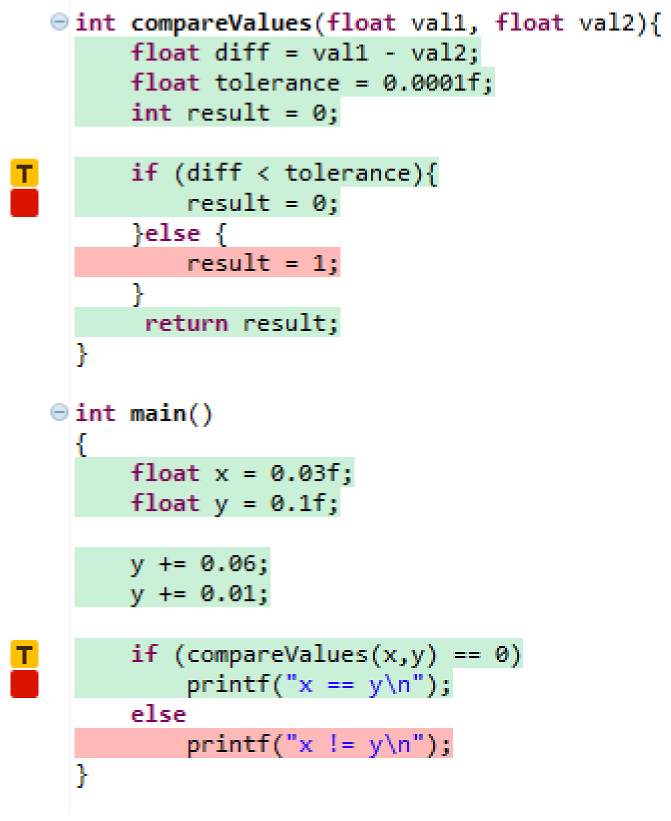
1.27. When comparing a variable of a floating point type, the result is different.

Incorrect results may be obtained when comparing floating-point type variables.



```
int main()
{
    float x = 0.03f;
    float y = 0.1f;

    y += 0.06;
    y += 0.01;

    if (x == y)
        printf("x == y\n");
    else
        printf("x != y\n");
}
```

The example is a known issue that occurs when comparing floating points. A typical example is the case where the result of comparing x and y is false.
This issue should not be viewed as a bug in the tool, but as an issue that discovered a bug that may occur when the tool does some floating-point comparison operations directly.



```
int compareValues(float val1, float val2){
    float diff = val1 - val2;
    float tolerance = 0.0001f;
    int result = 0;

    if (diff < tolerance){
        result = 0;
    }else {
        result = 1;
    }
     return result;
}

int main()
{
    float x = 0.03f;
    float y = 0.1f;

    y += 0.06;
    y += 0.01;

    if (compareValues(x,y) == 0)
        printf("x == y\n");
    else
        printf("x != y\n");
}
```

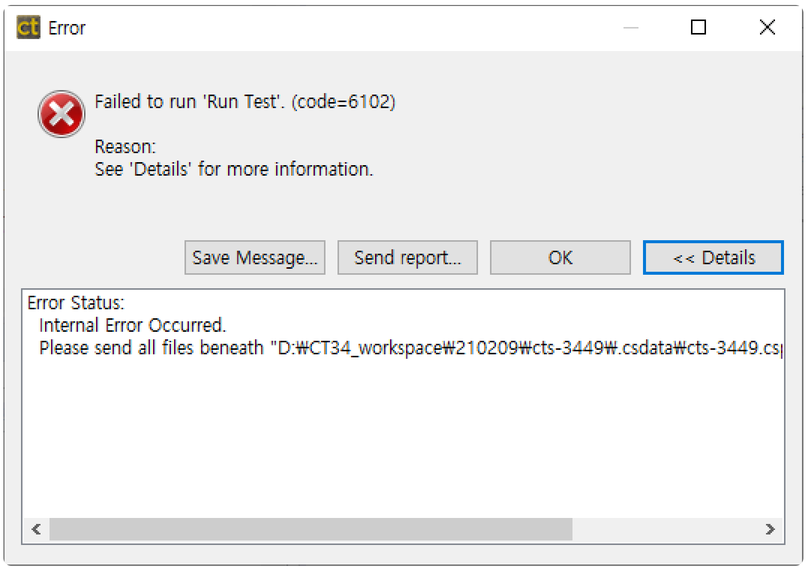
In this case, you can write a simple comparison function as in the example above and compare the values within the tolerance to solve it.

Please refer to the link below for more information.

https://randomascii.wordpress.com/2012/02/25/comparing-floating-point-numbers-2012-edition/
http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm
http://devmachine.blog.me/220119534107

1.28. When running tests, test execution fails after build is successful

When build is successful but execution fails during running tests, you can check *engine.log* if the following diolog appears and there is no error message.



Failed to run 'Run Test'. (code=6102)

Reason:
See 'Details' for more information.

```
Error Status:
  Internal Error Occurred.
  Please send all files beneath "D:₩CT34_workspace₩210209₩cts-3449₩.csdata₩cts-3449.cs
```

- *engine.log* path : `project_path\.csdata\log\engine.log`

If the following error message is displayed in the *engine.log* file, you can solve the problem by the following method.

```
[CEM] CFGFunction::load:not found function(qualifiedName, idFunctionInfo, func
tionKey)
```

1. Modify `TYPE_MAX_BUFFER_SIZE` value to a large value (ex. 5000) in the `project_path\.csdata\pa.ini` file.
2. After modifying `TYPE_MAX_BUFFER_SIZE` value, reanalyze the project and run the tests.

If there is no such error in *engine.log* or if it is not resolved by changing the TYPE_MAX_BUFFER_SIZE value, please contact technical support.

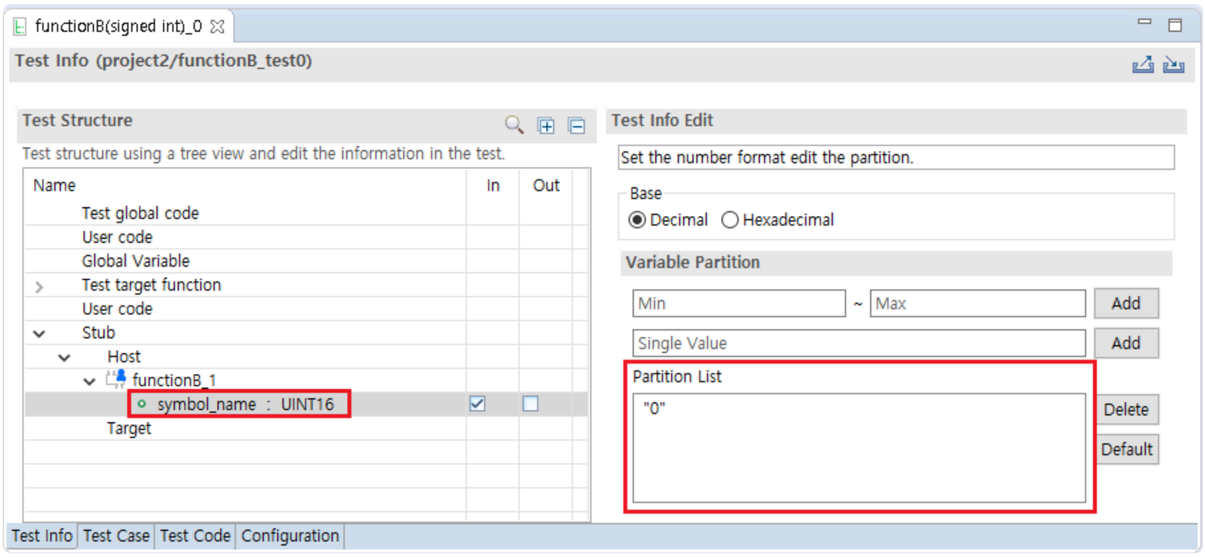
- Technical support contact : help@suresofttech.com

1.29. When only “0” exists in the partition list of the stub variable

The CS_XXX_INPUT macro can use a symbol by putting the type as the first argument and the symbol name as the second argument.
If you put a user-defined type other than the basic type (int float, unsigned short, etc.) in the first argument as follows,

```
CS_UINT_INPUT(UINT16, "symbol_name");
```

Since `UINT16` is not a basic type, only “0” exists because it cannot process partition information for that type.


functionB(signed int)_0

Test Info (project2/functionB_test0)

Test Structure

Test structure using a tree view and edit the information in the test.

Name	In	Out
Test global code		
User code		
Global Variable		
Test target function		
User code		
Stub		
Host		
functionB_1		
symbol_name : UINT16	✓	☐
Target		

Test Info Edit

Set the number format edit the partition.

Base: [✓ Decimal] Hexadecimal

Variable Partition

Min ~ Max — Add

Single Value — Add

Partition List

"0"

Delete

Default

Test Info | Test Case | Test Code | Configuration

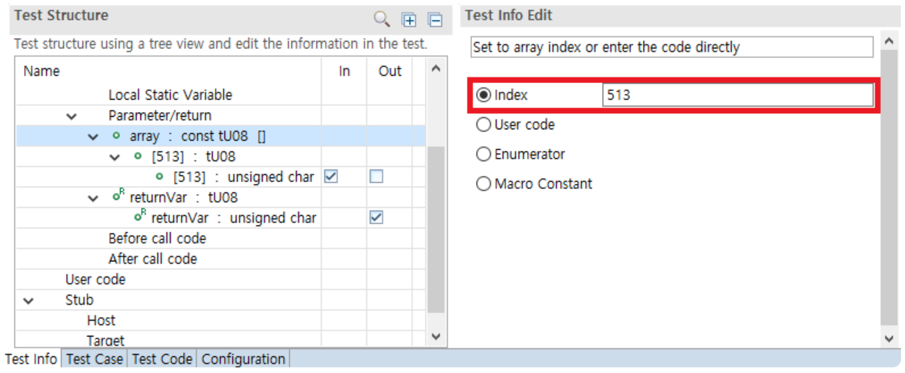
1.30. When stub input data information is deleted

After connecting the test and the stub, if you modify the symbol name or type in the CS macro, the previously edited stub-related input data will be deleted.
If the symbol name or type of the stub connected to the test is changed, the stub definition is read again and the previously read symbol is deleted to process I/O symbols.

1.31. When a signal error occurs in a function that takes a variable array as an argument

```
static tU08 getCheckSum(tU08 const array[])
{
    tU16  index;
    tU16 sum = 0U;
    tU08 checksum;
    for(index = 0; index < 511U; index++)
    {
        sum = sum + array[index];
    }
    checksum = (tU08)(sum & 0xffU);
    return checksum;
}
```

When executing the test of a function that takes a variable array as an argument as above, a signal error may occur.
In this case, you need to check whether you are accessing the array received as an argument in the function by index.
When accessing an index larger than the array length specified in the Test Info, you must adjust the index value in [Test Info] > [Test Structure]> [Test Info Edit] of variable array.


Test Structure
Test structure using a tree view and edit the information in the test.

Name	In	Out
Local Static Variable		
Parameter/return		
array : const tU08 []		
[513] : tU08		
[513] : unsigned char	✓	☐
returnVar : tU08		
returnVar : unsigned char		✓
Before call code		
After call code		
User code		
Stub		
Host		
Target		

Test Info Edit
Set to array index or enter the code directly
[✓ Index] 513
User code
Enumerator
Macro Constant

If you rerun the test after adjusting the index value, you can see that the signal error has disappeared.

You must adjust the index value to a value that is sufficiently larger than the index value accessed within the function to obtain normal execution results.

1.32. When the size of the structure is different from Controller Tester and in the original program

```
#pragma pack(1)
typedef struct TEST_STRUCTURE {
   /* Byte 0 */
     UINT8 uiSEQ;
   /* Byte 1 */
     BIT1 biCharDir :1;
     UINT32 uiReserved1 :7;
   /* Byte 2 */
     UINT8 uiPulseCount_Ch1;
   /* Byte 3 */
     UINT8 uiPulseCount_Ch2;
   /* Byte 4~7 */
     UINT32 uiPulseCount_Access;
   /* Byte 8 */
     UINT32 uiSwVer :6;
     UINT32 uiStatus :2;
   /* Byte 9~11 */
     UINT16 uiSizeIndex;
     UINT8 uiReserved2;
   /* Byte 12~15 */
     UINT32 uiC;
};
```

In the case of code including the `#pragma pack` directive as above, the size of the structure in Controller Tester may be displayed differently from the original program.
The above directive is a structure memory alignment option, and pack support and behavior can differ depending on the compiler.
Therefore, in the case of using the conversion toolchain in Controller Tester, you can solve the above problem by selecting the visual studio compiler or gcc compiler that matches the compiler used to build the original program and the pack behavior.

1.33. When an undefined referenced error occurs in the IAR environment

This can happen when anonymous union or anonymous struct is used in IAR environment.
In the IAR compiler, member variables of anonymous union or anonymous struct can be accessed and used like global variables.
Due to this, an error occurs in which certain variables cannot be found when analyzing in Controller Tester.
Member variables of anonymous union and anonymous struct must be added as global variables in order to save the meaning of the special syntax of IAR.

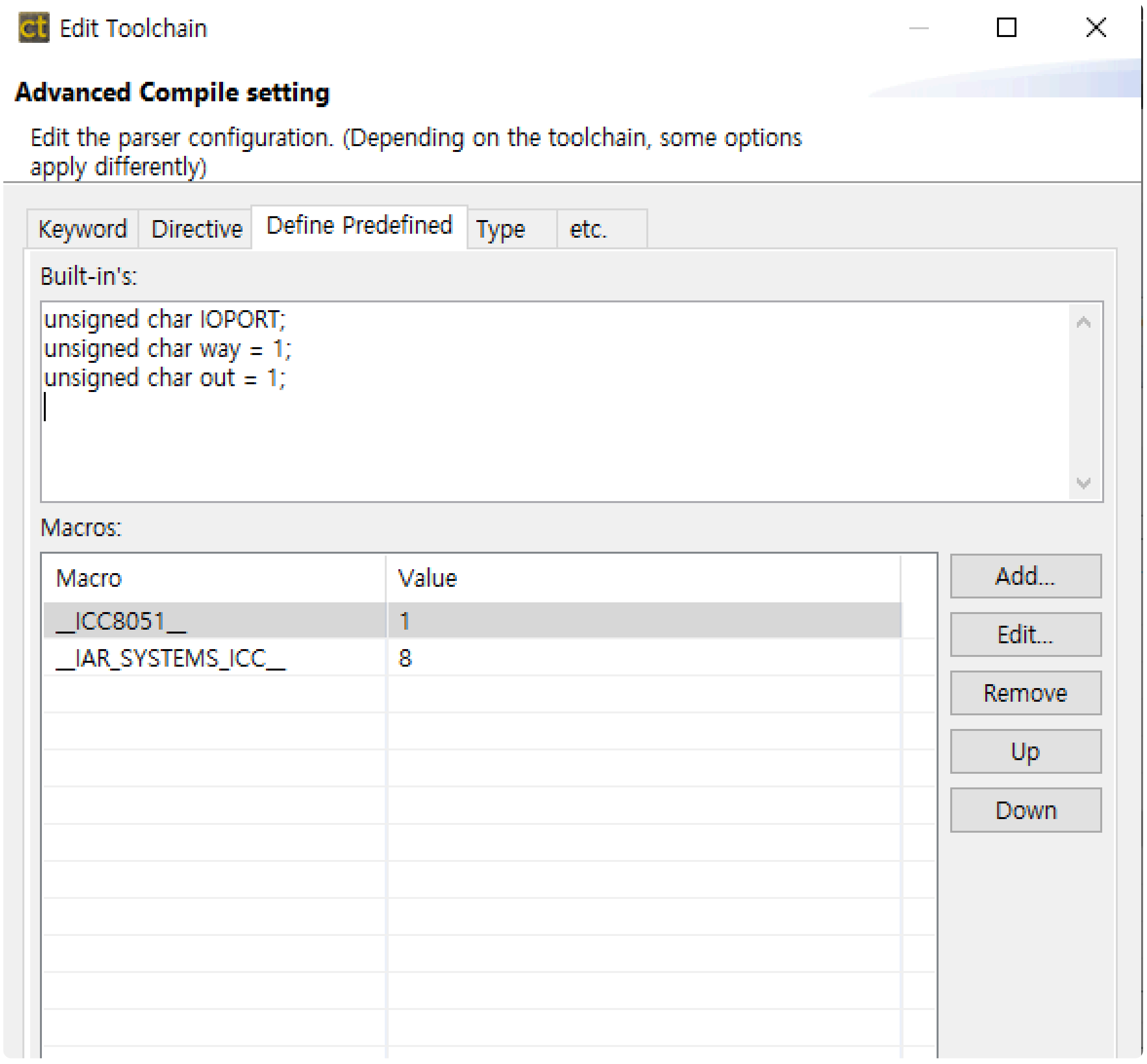
How to add a global variable is as follows.

- ‘Preferences’ > ‘Toolchain’ > ‘edit’ After selecting the toolchain > ‘Predefined declaration’ > Add variable declaration to ‘Built-in declaration’

Example of using IAR anonymous union

```
__no_init volatile
union
{
 unsigned char IOPORT;
 struct
 {
 unsigned char way: 1;
 unsigned char out: 1;
 };
} @ 0x1000;
/* The variables are used here. */
void Test(void)
{
 IOPORT = 0; // Use union member variables like global variables
 way = 1; // Use struct member variables like global variables
 out = 1; // Use struct member variables like global variables
}
```

Global variable added to the toolchain



Advanced Compile setting

Edit the parser configuration. (Depending on the toolchain, some options apply differently)

Built-in's:

```
unsigned char IOPORT;
unsigned char way = 1;
unsigned char out = 1;
```

Macros:

Macro	Value
__ICC8051__	1
__IAR_SYSTEMS_ICC__	8

1.34. When an error occurs in the test code due to the existence of a const global variable

First you can remove the const of the global variable by converting the option below in your toolchain.ini file.

```
global_variable_nonconstant = 1
```

However, it is impossible to compile the test code that is generated by default when the test is executed, so an error occurs in the converting stage.

If you modify the test code into a compilable form and run the test, you can build normally.

```
void Rte_MemCpy(void *destination, void *source, unsigned long num);

typedef unsigned char CanMsgData[8];
typedef unsigned char uint8;
CanMsgData data;
const CanMsgData Rte_C_CanMsgData_0 = {
  0U, 0U, 0U, 0U, 0U, 0U, 0U, 0U
};

void Rte_MemCpy(void *destination, void *source, unsigned long num)
{
}
void main(void)
{
 Rte_MemCpy(data, Rte_C_CanMsgData_0, sizeof(CanMsgData));
}
```

In the code above, the test code generated by default is as follows.

```
/*Input*/
Rte_C_CanMsgData_0[0] = CS_UINT_INPUT(unsigned char,"Rte_C_CanMsgData_0[0]");
data[0] = CS_UINT_INPUT(unsigned char,"data[0]");
```

Since the Rte_C_CanMsgData_0 is declared as a const global variable, an error occurs.
`error: expression must be a modifiable lvalue`

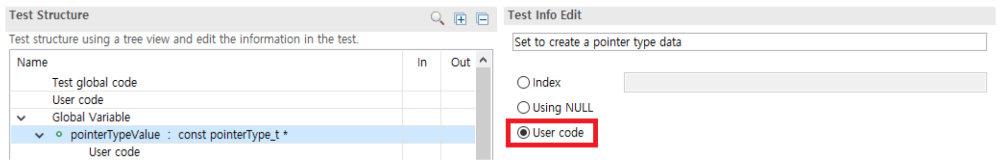
Change the test information of the global variable into user code and modify it in a compilable form as follows.

```
unsigned char * Rte_C_CanMsgData_Temp = (unsigned char *)Rte_C_CanMsgData_0;
// If it points to an address, const cannot be determined at compile time, so
it is judged to be a code that compiles normally in conversion and operates no
rmally
// When checking const at runtime, the const is already removed due to the glo
bal_variable_nonconstant option, so a runtime error does not occur.
Rte_C_CanMsgData_Temp[0] = CS_INT_INPUT(unsigned char, "Rte_C_CanMsgData_Tem
p[0]");
Rte_C_CanMsgData_Temp[1] = CS_INT_INPUT(unsigned char, "Rte_C_CanMsgData_Tem
p[1]");
Rte_C_CanMsgData_Temp[2] = CS_INT_INPUT(unsigned char, "Rte_C_CanMsgData_Tem
p[2]");
Rte_C_CanMsgData_Temp[3] = CS_INT_INPUT(unsigned char, "Rte_C_CanMsgData_Tem
p[3]");
Rte_C_CanMsgData_Temp[4] = CS_INT_INPUT(unsigned char, "Rte_C_CanMsgData_Tem
p[4]");
```

1.35. When a signal error occurs when accessing a static const pointer variable

```
static const pointerType_t* pointerTypeValue;

extern pointerType_t* externPointerTypeFunction(void);

static void getMessageFunction(tU08 timeTick){
        MessageInfo msg;
        tS08 tick= (tS08)timeTick;

        while(tick >= 0){
                tick = tick-4;
                if(tick==0){
                        msg = (*pointerTypeValue->message)(); // signal error
occurred
                        break;
                }
        }
}
```

When accessing a static const pointer variable, a signal error may occur if the static const pointer variable is not initialized.
In this case, open the test editor of the function in Controller Tester, find the uninitialized static const pointer variable in the test structure tree, and designate the data generation method as [user code].



Test Structure
Test structure using a tree view and edit the information in the test.

Name	In	Out
Test global code		
User code		
Global Variable		
pointerTypeValue : const pointerType_t *		
User code		

Test Info Edit
Set to create a pointer type data

Index
Using NULL
[✓ User code]

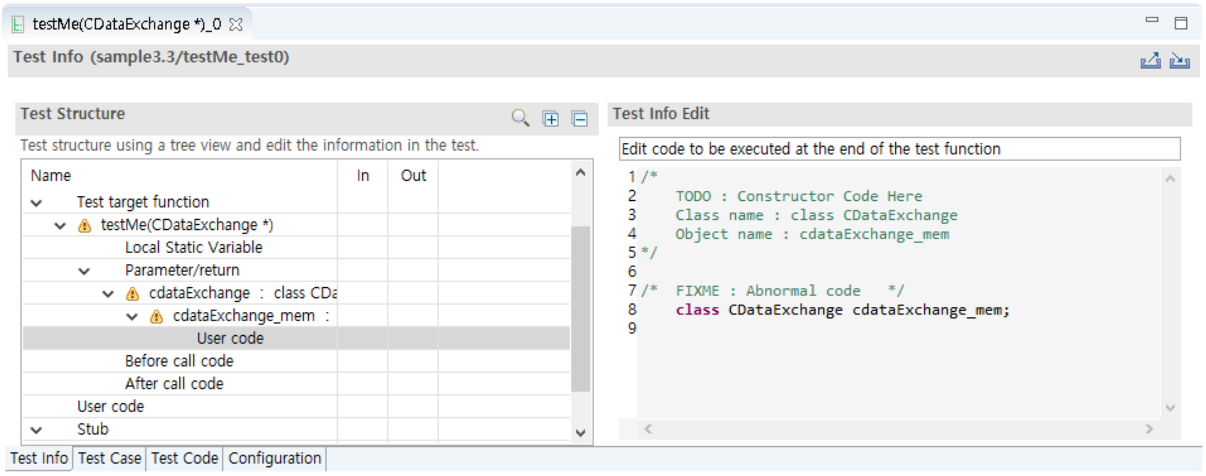
After initializing a new object of the same type in the user code editor, make the static const variable declared in the code point to it.
Below is an example of the user code for the code above.

```
pointerType_t temp1 = {0x0,0x0,externPointerTypeFunction, };
pointerType_t* temp2 = &temp1;
pointerTypeValue = temp2;
```

You can solve the signal error by saving the test editor and re-running the test.

1.36. “C2512: No suitable default constructor available.” If an error occurs

```
#include <afxwin.h>
void testMe(CDataExchange *cdataExchange)
{
return;
}
```


testMe(CDataExchange *)_0

Test Info (sample3.3/testMe_test0)

Test Structure

Test structure using a tree view and edit the information in the test.

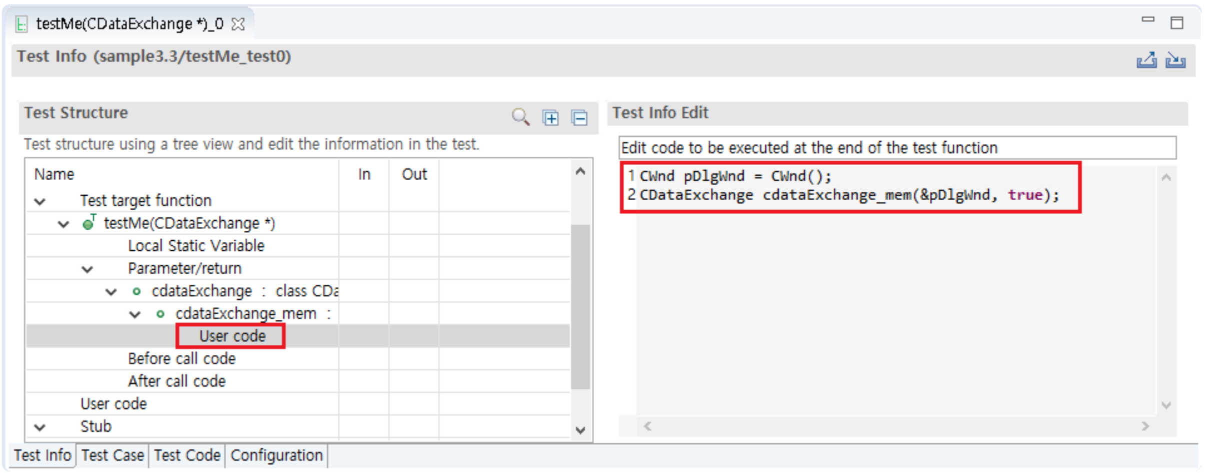
Name	In	Out
Test target function		
testMe(CDataExchange *)		
Local Static Variable		
Parameter/return		
cdataExchange : class CDa		
cdataExchange_mem :		
User code		
Before call code		
After call code		
User code		
Stub		

Test Info Edit

Edit code to be executed at the end of the function

```
/*
    TODO : Constructor Code Here
    Class name : class CDataExchange
    Object name : cdataExchange_mem
*/

/*  FIXME : Abnormal code   */
    class CDataExchange cdataExchange_mem;
```

If a constructor of a class that is not analyzed in Controller Tester is used, the default constructor is automatically selected.
If the above example is run without modification, the default constructor is chosen, resulting in “error C2512: No suitable default constructor available.” It prints a message and a compilation error occurs.


testMe(CDataExchange *)_0

Test Info (sample3.3/testMe_test0)

Test Structure

Test structure using a tree view and edit the information in the test.

Name	In	Out
Test target function		
testMe(CDataExchange *)		
Local Static Variable		
Parameter/return		
cdataExchange : class CDa		
cdataExchange_mem :		
User code		
Before call code		
After call code		
User code		
Stub		

Test Info Edit

Edit code to be executed at the end of the function

```
CWnd pDlgWnd = CWnd();
CDataExchange cdataExchange_mem(&pDlgWnd, true);
```

In this case, as in the example above, you can solve the problem by directly entering the appropriate constructor of the class in the user code.

! Since CT3.4 version and later, a default constructor is created for some classes, so this problem may not occur.

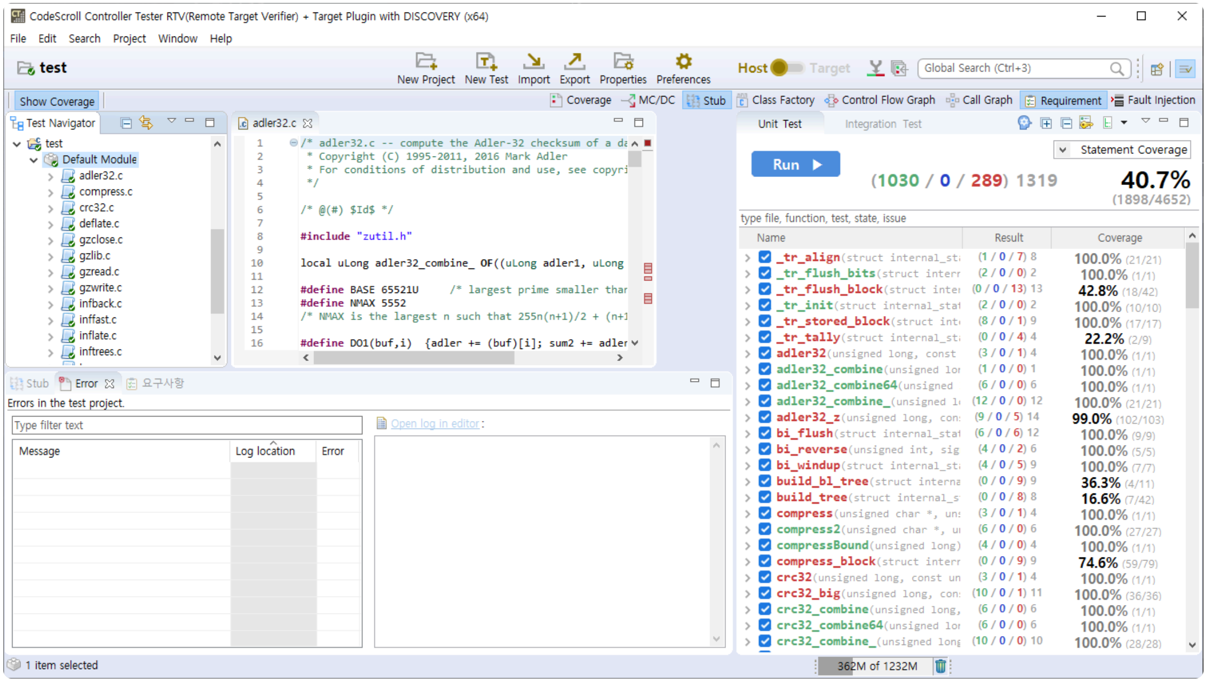
1.37. How to design stub code that behaves differently for each test case

When running the test after adding the stub to the test, the execution result can be different for each test case.
The following is an example of stub code in which the result returned by the stub is written differently for each test case.

```
int result = 0;
if(CS_TESTCASENO() == 1) { // Modified to allow different behavior for each te
st case
        result = some_function1();
} else if(CS_TESTCASENO() == 2) {
        result = some_function2();
}
return result;
```

1.38. When perspective of Controller Tester is broken

When testing is carried out by moving views in Controller Tester, the perspective may become strange as shown in the figure below.


CodeScroll Controller Tester RTV(Remote Target Verifier) + Target Plugin with DISCOVERY (x64)
File Edit Search Project Window Help
test
New Project New Test Import Export Properties Preferences
Host Target
Global Search (Ctrl+3)
Show Coverage
Coverage MC/DC Stub Class Factory Control Flow Graph Call Graph Requirement Fault Injection
Test Navigator
test
Default Module
adler32.c
compress.c
crc32.c
deflate.c
gzclose.c
gzlib.c
gzread.c
gzwrite.c
infback.c
inffast.c
inflate.c
inftrees.c
adler32.c
Stub Error 요구사항
Errors in the test project.
Type filter text
Message Log location Error
Open log in editor :
1 item selected
Unit Test Integration Test
Statement Coverage
Run
type file, function, test, state, issue
362M of 1232M

```
/* adler32.c -- compute the Adler-32 checksum of a da
 * Copyright (C) 1995-2011, 2016 Mark Adler
 * For conditions of distribution and use, see copyri
 */

/* @(#) $Id$ */

#include "zutil.h"

local uLong adler32_combine_ OF((uLong adler1, uLong

#define BASE 65521U     /* largest prime smaller thar
#define NMAX 5552
/* NMAX is the largest n such that 255n(n+1)/2 + (n+1

#define DO1(buf,i)  {adler += (buf)[i]; sum2 += adler
```

(1030 / 0 / 289) 1319 40.7% (1898/4652)

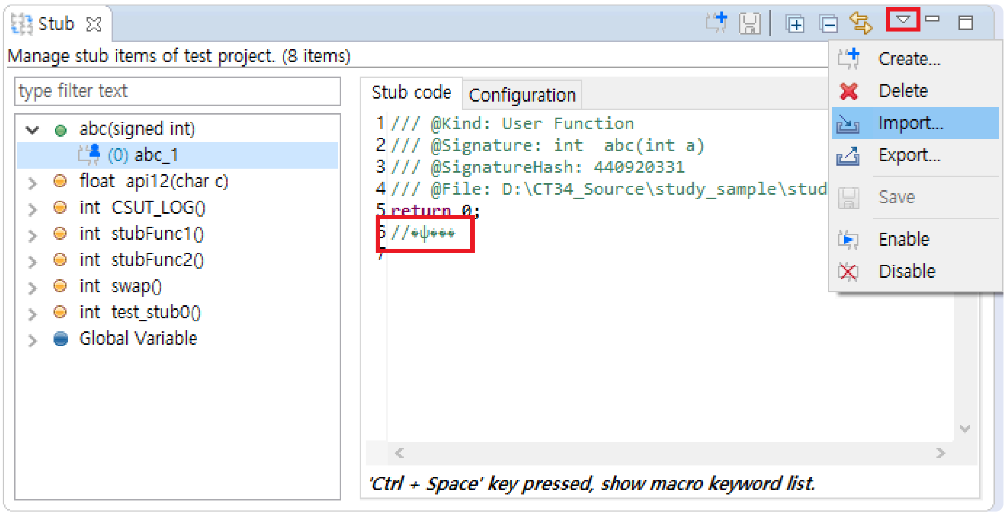
Name	Result	Coverage
_tr_align(struct internal_st	(1 / 0 / 7) 8	100.0% (21/21)
_tr_flush_bits(struct interr	(2 / 0 / 0) 2	100.0% (1/1)
_tr_flush_block(struct inter	(0 / 0 / 13) 13	42.8% (18/42)
_tr_init(struct internal_stat	(2 / 0 / 0) 2	100.0% (10/10)
_tr_stored_block(struct int	(8 / 0 / 1) 9	100.0% (17/17)
_tr_tally(struct internal_st	(0 / 0 / 4) 4	22.2% (2/9)
adler32(unsigned long, const	(3 / 0 / 1) 4	100.0% (1/1)
adler32_combine(unsigned lor	(1 / 0 / 0) 1	100.0% (1/1)
adler32_combine64(unsigned	(6 / 0 / 0) 6	100.0% (1/1)
adler32_combine_(unsigned l	(12 / 0 / 0) 12	100.0% (21/21)
adler32_z(unsigned long, con	(9 / 0 / 5) 14	99.0% (102/103)
bi_flush(struct internal_stat	(6 / 0 / 6) 12	100.0% (9/9)
bi_reverse(unsigned int, sig	(4 / 0 / 2) 6	100.0% (5/5)
bi_windup(struct internal_st	(4 / 0 / 5) 9	100.0% (7/7)
build_bl_tree(struct interna	(0 / 0 / 9) 9	36.3% (4/11)
build_tree(struct internal_s	(0 / 0 / 8) 8	16.6% (7/42)
compress(unsigned char *, uns	(3 / 0 / 1) 4	100.0% (1/1)
compress2(unsigned char *, ur	(6 / 0 / 0) 6	100.0% (27/27)
compressBound(unsigned long)	(4 / 0 / 0) 4	100.0% (1/1)
compress_block(struct interr	(0 / 0 / 9) 9	74.6% (59/79)
crc32(unsigned long, const un	(3 / 0 / 1) 4	100.0% (1/1)
crc32_big(unsigned long, con	(10 / 0 / 1) 11	100.0% (36/36)
crc32_combine(unsigned long,	(6 / 0 / 0) 6	100.0% (1/1)
crc32_combine64(unsigned lor	(6 / 0 / 0) 6	100.0% (1/1)
crc32_combine_(unsigned long	(10 / 0 / 0) 10	100.0% (28/28)

You can restore the perspective via [Window] – [Reset Perspective…]. If [Reset Perspective…] does not solve the problem, reset the perspective using the method below.

1. Quit Controller Tester.
2. Delete the `workspace_path\.metadata\.plugins\org.eclipse.e4.workbench\workbench.xmi` file.
3. Restart Controller Tester.

1.39. Problem that Korean comments are broken when exporting stub code before Controller Tester 3.4 and importing after 3.4

In Stub view, when exporting stub code before Controller Tester 3.4 and importing stub code before Controller Tester 3.4 after Controller Tester 3.4, Korean comments are broken.



```
/// @Kind: User Function
/// @Signature: int  abc(int a)
/// @SignatureHash: 440920331
/// @File: D:\CT34_Source\study_sample\stud
return 0;
//◆ψ◆◆◆
```

This is because the encoding of the stub code is ANSI until Controller Tester 3.3 and UTF-8 after Controller Tester 3.4. If you import stub code that was exported before Controller Tester 3.4, you must use an external editor to change the encoding to UTF-8 before importing.

1.40. When a value cannot be entered in the test case because the register variable is declared as a macro

```
#define IN_IG_ON (int8_t)(PORTDbits.RD0)
#define IN_INTER_LOCK_SNSR (int8_t)(PORTDbits.RD2)
#define IN_DIFF_LOCK_SNSR_2ND (int8_t)(PORTDbits.RD3)
#define IN_DIFF_LOCK_SNSR_3RD (int8_t)(PORTDbits.RD12)

if((IN_IG_ON == OFF) && (getHazardSwitch() == OFF) && (getBrakeSwitch() == OF
F))
{

}
```

In the code above, the return value of the getHazardSwitch() or getBrakeSwitch() function can be changed using a stub, but the value of IN_IG_ON is declared as a macro, so the value cannot be controlled in the test case.

In this case, the customer needs to modify the code so that the use of register variables can be bundled into a function and treated as a stub.

1.41. How to convert a report in xls format to html format

You can run the script to convert the xls formatted report exported from CT to html format.

1. After decompressing the xls_to_html.zip file, open the xls_to_html.vbs file with an editor.
2. In ctTestResultXLS, enter the directory path of the report to be converted.
3. In ctTestResultHTML, enter the directory path where the report in html format will be generated.

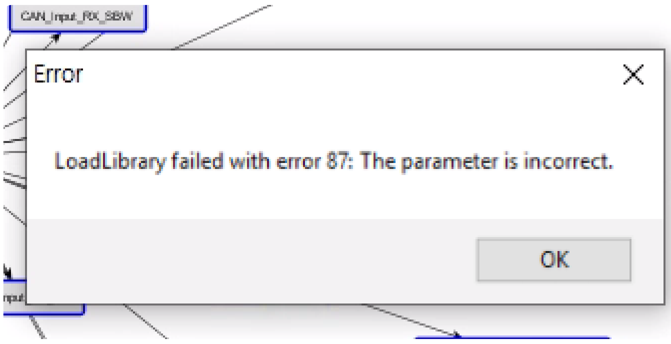
! The directory path must have been created in advance, and must include \ at the end of the path.

4. Run xls_to_html.bat from cmd or type Start xls_to_html.vbs to run the script

xls_to_html.zip

1.42. “LoadLibrary failed with error 87: The parameter is incorrect.” error occurs

When using Controller Tester, sometimes a warning window like “LoadLibrary failed with error 87: The parameter is incorrect.” appears.



LoadLibrary failed with error 87: The parameter is incorrect.

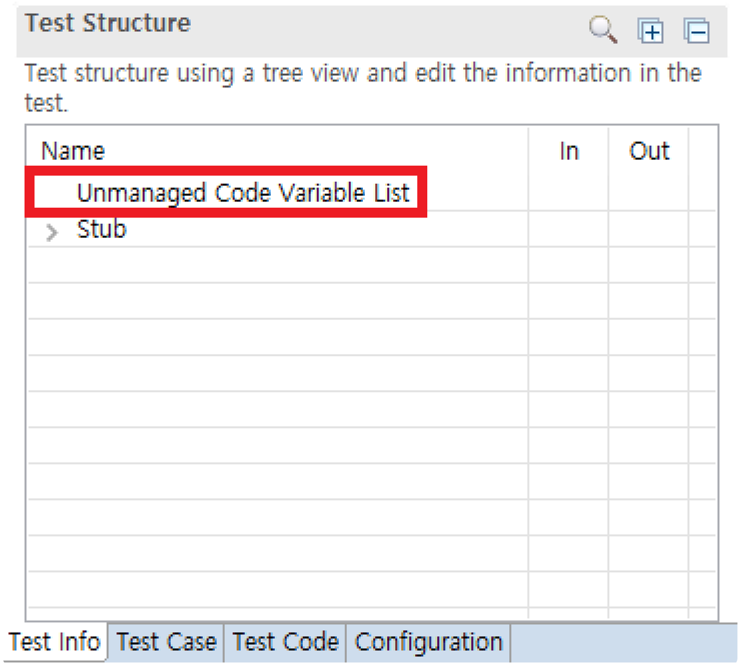
This error occurs when there is a problem with the graphics card driver installed on the PC, or there is a conflict between the graphics card of the host PC and the remote PC when using remote access.
You can try the following methods to solve the problem.

- Update your graphics card driver to the latest version.
- If the current driver is up to date, revert to the previous version.
- If more than one graphics card is turned on, disable the graphics card in problem.
- If using remote access, disable the graphics card of the host PC and try remote access.
- Completely uninstall Controller Tester and reinstall.

1.43. Unmanaged code variable list cannot be displayed in the test editor (for versions before CT 3.4)

In Controller Tester 3.3 or earlier, the variable list may not be displayed in the test information window of the test editor in the following cases.

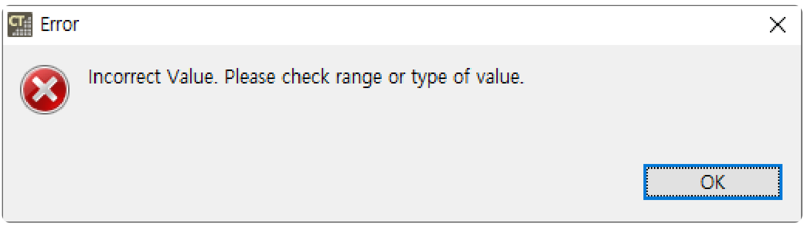
- When many variables are added using input/output macros in unmanaged test code or stub code
- When execute [Import test] or [Import test using test code file] of the above type of test
- When saving the test code or stub code in the form above after some modifications


Test Structure
Test structure using a tree view and edit the information in the test.
Name
In
Out
Unmanaged Code Variable List
Stub
Test Info
Test Case
Test Code
Configuration

This is because Controller Tester 3.3 and earlier limit the number of variables that can be expressed in the test information window to a maximum of 4096.
If you need to use more variables than the limit, please upgrade Controller Tester to version 3.4 or higher or contact the support team(help@suresofttech.com).

1.44. How to enter invalid values in test data

In Controller Tester, an error occurs if you enter an invalid value for input value or expected value of the test case.


Error

Incorrect Value. Please check range or type of value.

- Ex 1. When entering a negative value into a `unsigned int` type variable.
- Ex 2. When entering a value less than -128 or greater than 127 in a `char` type variable.

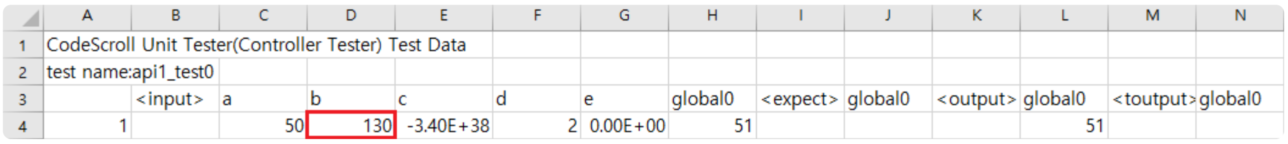
How to enter invalid values in test data

The variable used in the example is `b` and is of type `char`.

1. In Test view, right-click the function to enter an invalide value and export test data.
2. Open the exported test data file(.csv).
3. Enter values according to the file format and save.

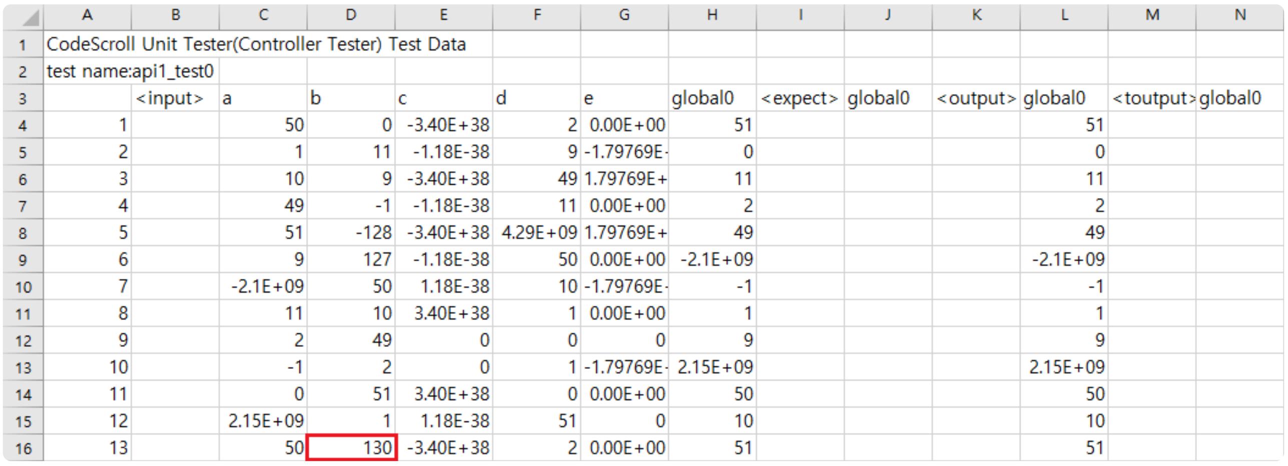
After Controller Tester 3.5, the policy of importing test data has been changed into adding test cases. If you only want to import data for a specific test case, please delete the existing test case before importing.

- After Controller Tester 3.5



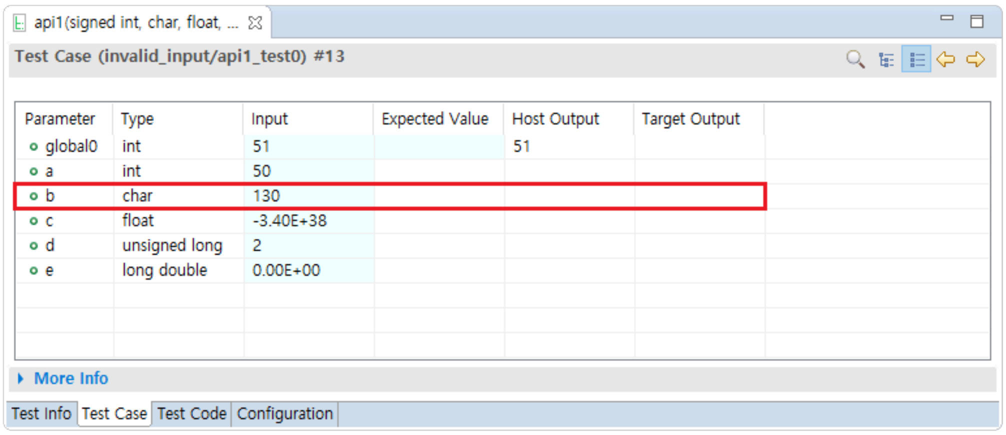
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CodeScroll Unit Tester(Controller Tester) Test Data													
2	test name:api1_test0													
3		<input>	a	b	c	d	e	global0	<expect>	global0	<output>	global0	<toutput>	global0
4	1		50	130	-3.40E+38	2	0.00E+00	51				51		

- Before Controller Tester 3.4



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CodeScroll Unit Tester(Controller Tester) Test Data													
2	test name:api1_test0													
3		<input>	a	b	c	d	e	global0	<expect>	global0	<output>	global0	<toutput>	global0
4	1		50	0	-3.40E+38	2	0.00E+00	51				51		
5	2		1	11	-1.18E-38	9	-1.79769E-	0				0		
6	3		10	9	-3.40E+38	49	1.79769E+	11				11		
7	4		49	-1	-1.18E-38	11	0.00E+00	2				2		
8	5		51	-128	-3.40E+38	4.29E+09	1.79769E+	49				49		
9	6		9	127	-1.18E-38	50	0.00E+00	-2.1E+09				-2.1E+09		
10	7		-2.1E+09	50	1.18E-38	10	-1.79769E-	-1				-1		
11	8		11	10	3.40E+38	1	0.00E+00	1				1		
12	9		2	49	0	0	0	9				9		
13	10		-1	2	0	1	-1.79769E-	2.15E+09				2.15E+09		
14	11		0	51	3.40E+38	0	0.00E+00	50				50		
15	12		2.15E+09	1	1.18E-38	51	0	10				10		
16	13		50	130	-3.40E+38	2	0.00E+00	51				51		

4. In Test view, right-click the function and import the saved test data.
5. In Test Case tab of Test Editor, you can see the test data that you import.



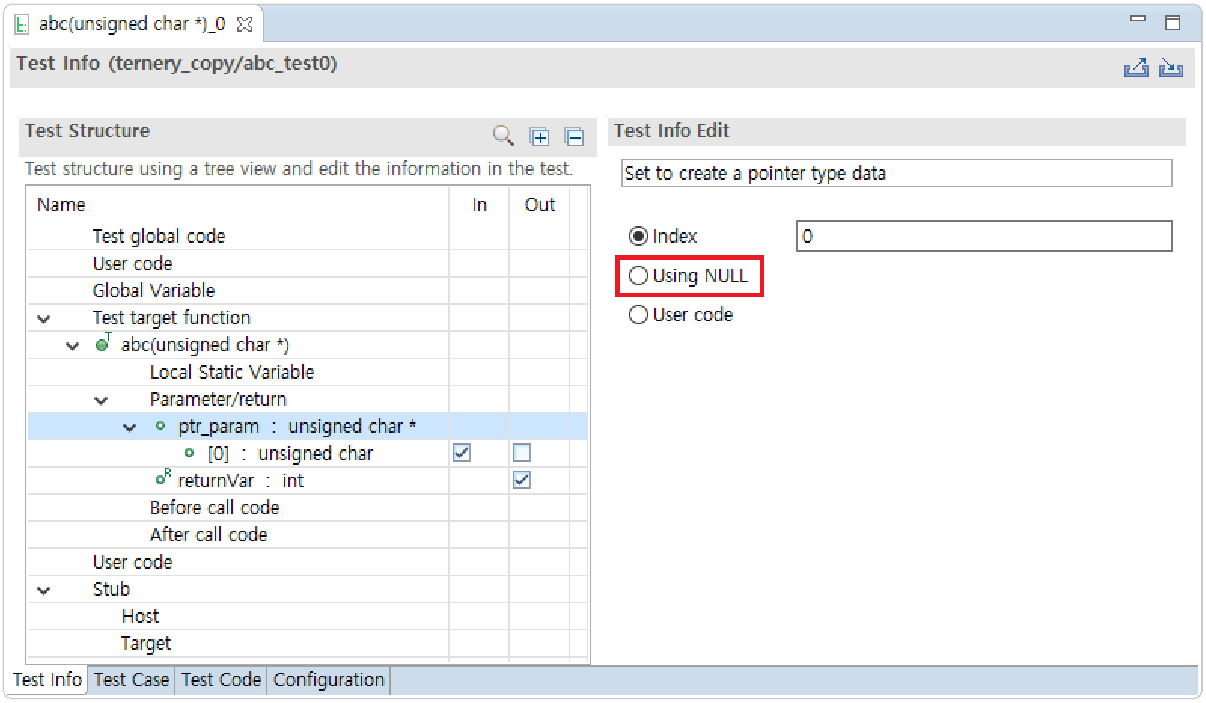
Test Case (invalid_input/api1_test0) #13

Parameter	Type	Input	Expected Value	Host Output	Target Output
global0	int	51		51	
a	int	50			
b	char	130			
c	float	-3.40E+38			
d	unsigned long	2			
e	long double	0.00E+00			

1.45. How to enter a NULL value for a pointer-type parameter

How to enter a NULL value for a pointer-type parameter in tests

In [Test Editor] – [Test Info] tab – [Test Structure] tree, after selecting the appropriate parameters, select [Using NULL] in Test Info Edit and save.


abc(unsigned char *)_0
Test Info (ternery_copy/abc_test0)
Test Structure
Test structure using a tree view and edit the information in the test.
Test Info Edit
Test Info
Test Case
Test Code
Configuration

Name	In	Out
Test global code		
User code		
Global Variable		
Test target function		
abc(unsigned char *)		
Local Static Variable		
Parameter/return		
ptr_param : unsigned char *		
[0] : unsigned char	✓	☐
returnVar : int		✓
Before call code		
After call code		
User code		
Stub		
Host		
Target		

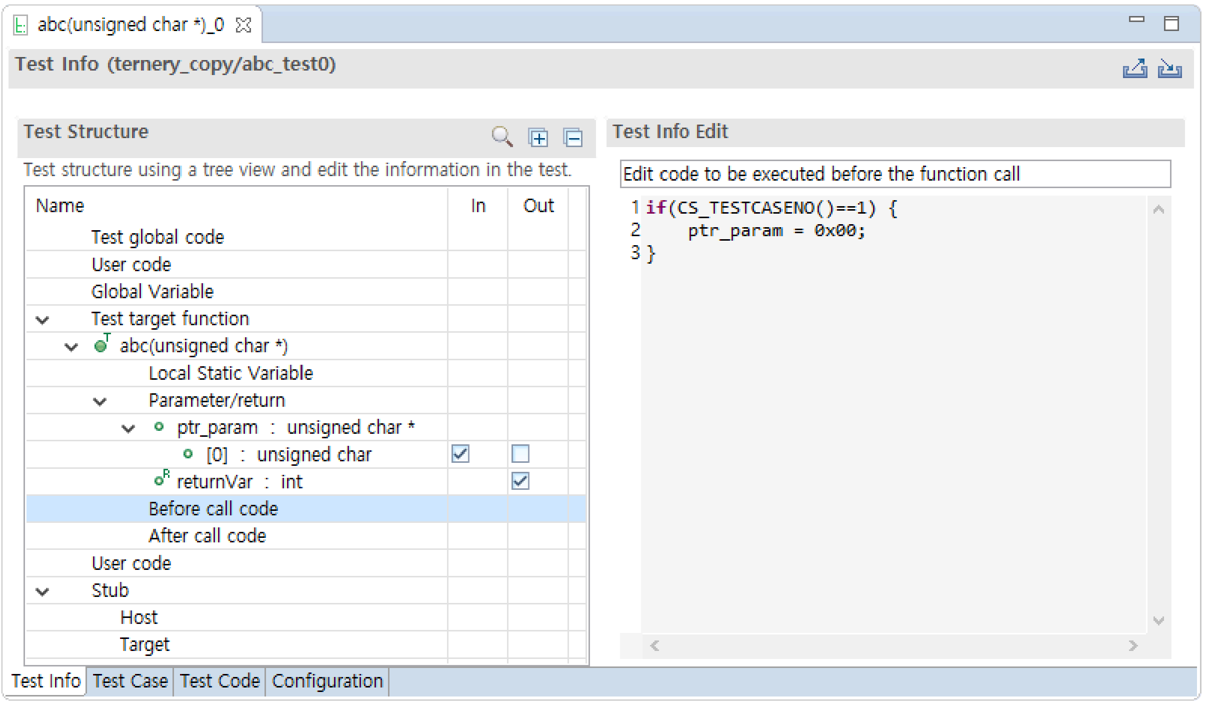
Set to create a pointer type data

[✓ Index] 0
Using NULL
User code

How to enter a NULL value for a pointer-type parameter in specific test cases

You can enter NULL values in a pointer-type parameter using test macro(`CS_TESTCASENO()`) that return test case number.

1. In [Test Editor] – [Test Info] tab – [Test Structure] tree, select one of the followings:
   - [Before call code]
   - [User code] in Test info Edit, after selecting the appropriate parameters
2. Enter the code to pass a NULL pointer as an argument in a specific test case.
   - The example code is the code to test by passing a NULL pointer as an argument when the test case is number 1.


abc(unsigned char *)_0

Test Info (ternery_copy/abc_test0)

Test Structure

Test structure using a tree view and edit the information in the test.

Name	In	Out
Test global code		
User code		
Global Variable		
Test target function		
abc(unsigned char *)		
Local Static Variable		
Parameter/return		
ptr_param : unsigned char *		
[0] : unsigned char	✓	☐
returnVar : int		✓
Before call code		
After call code		
User code		
Stub		
Host		
Target		

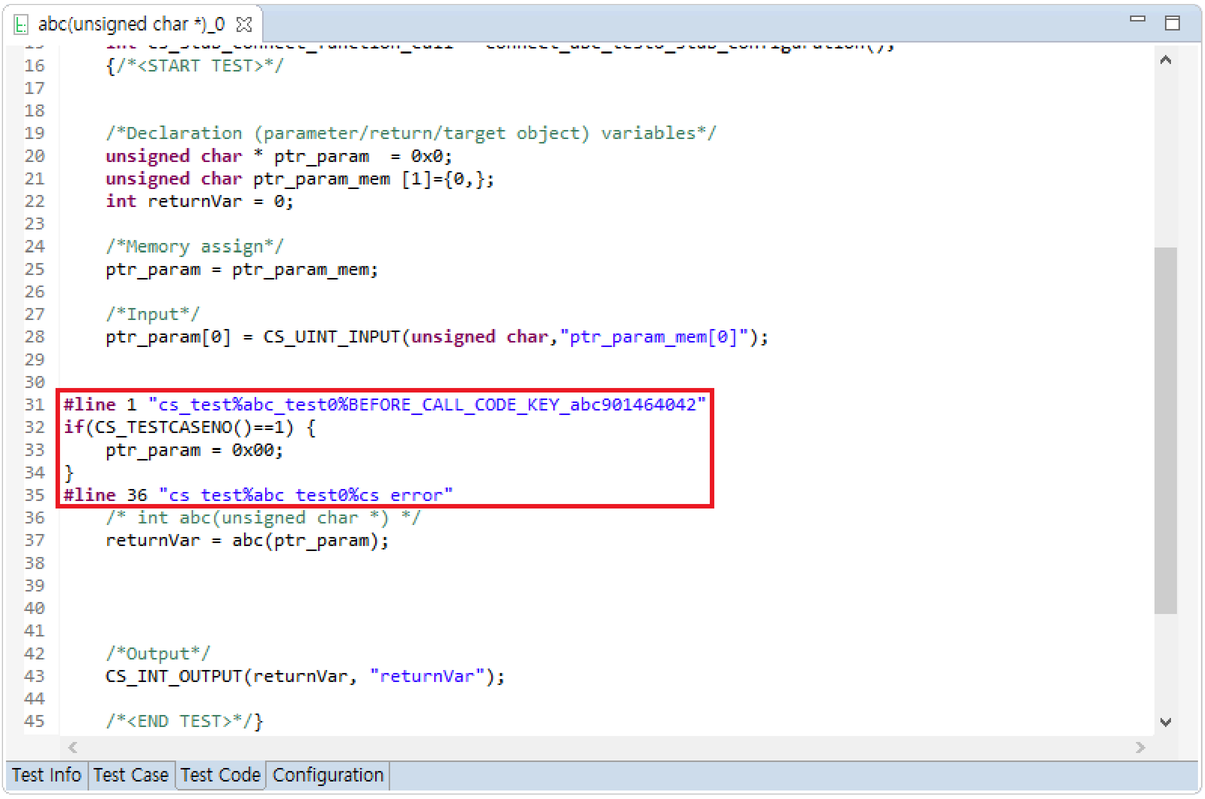
Test Info Edit

Edit code to be executed before the function call

```
if(CS_TESTCASENO()==1) {
    ptr_param = 0x00;
}
```

Test Info | Test Case | Test Code | Configuration

3. You can check the entered code in the [Test Code] tab. Just check the test code and enter the code appropriately.

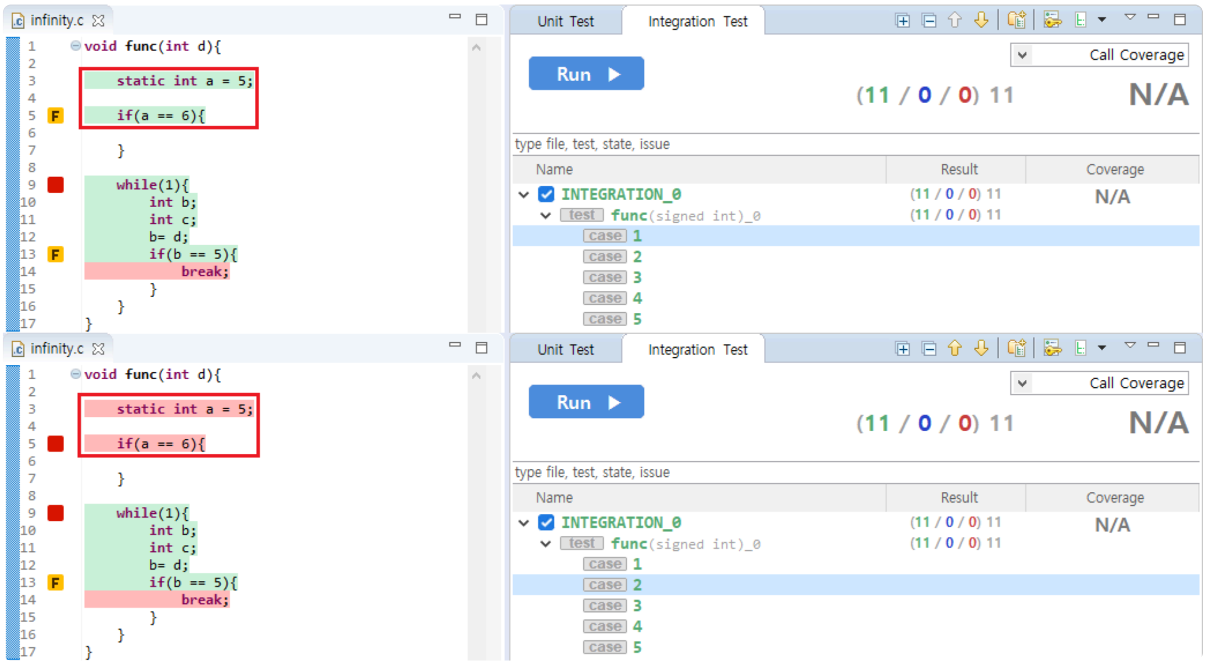

abc(unsigned char *)_0

```
    {/*<START TEST>*/


    /*Declaration (parameter/return/target object) variables*/
    unsigned char * ptr_param  = 0x0;
    unsigned char ptr_param_mem [1]={0,};
    int returnVar = 0;

    /*Memory assign*/
    ptr_param = ptr_param_mem;

    /*Input*/
    ptr_param[0] = CS_UINT_INPUT(unsigned char,"ptr_param_mem[0]");


#line 1 "cs_test%abc_test0%BEFORE_CALL_CODE_KEY_abc901464042"
if(CS_TESTCASENO()==1) {
    ptr_param = 0x00;
}
#line 36 "cs_test%abc_test0%cs_error"
    /* int abc(unsigned char *) */
    returnVar = abc(ptr_param);




    /*Output*/
    CS_INT_OUTPUT(returnVar, "returnVar");

    /*<END TEST>*/}
```

Test Info | Test Case | Test Code | Configuration

1.46. When using the conversion toolchain, the coverages are displayed abnormally in the integration test with the infinite loop removed.

When you use the conviersion toolchain to run host tests, if you set `remove_infinity_loop = 1` in the `toolchain.ini` file to remove the infinite loop and run the integration test, the coverage of the first test case is displayed normally, but the coverage starting from second test case is displayed abnormally.



```
void func(int d){

    static int a = 5;

    if(a == 6){

    }

    while(1){
        int b;
        int c;
        b= d;
        if(b == 5){
            break;
        }
    }
}
```

(11 / 0 / 0) 11   N/A

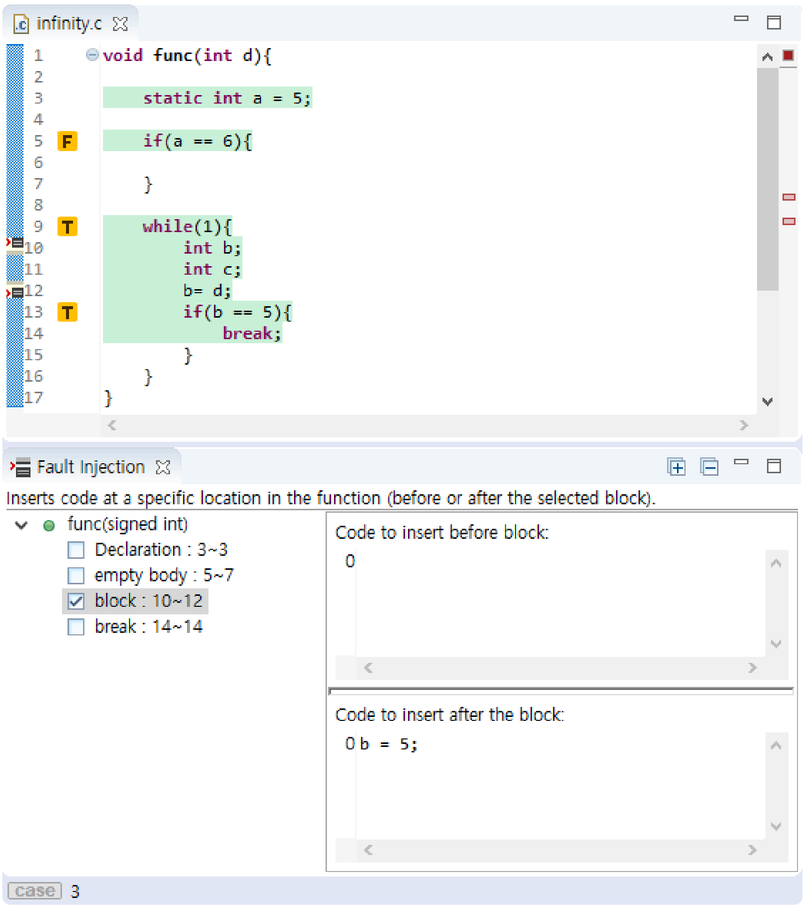
Name	Result	Coverage
INTEGRATION_0	(11 / 0 / 0) 11	N/A
test func(signed int)_0	(11 / 0 / 0) 11	
case 1		
case 2		
case 3		
case 4		
case 5		

This feature is provided by Controller Tester so that an infinite loop of the form `while(1)` can be repeated as many as the number of test cases in an integration test.

When users want to eliminate infinite loops in integration tests, they should use the fault injection feature.

1. Set `remove_infinity_loop = 0` in the `toolchain.ini` file.

2. Break out of the loop by injecting a fault inside the loop.
    - When a conditional statement that can break out of the infinite loop is inside the loop
        - Inject the fault appropriately so that can break out of the infinite loop through that condition statement.



```
void func(int d){

    static int a = 5;

    if(a == 6){

    }

    while(1){
        int b;
        int c;
        b= d;
        if(b == 5){
            break;
        }
    }
}
```

Inserts code at a specific location in the function (before or after the selected block).

- func(signed int)
  - ☐ Declaration : 3~3
  - ☐ empty body : 5~7
  - ☑ block : 10~12
  - ☐ break : 14~14

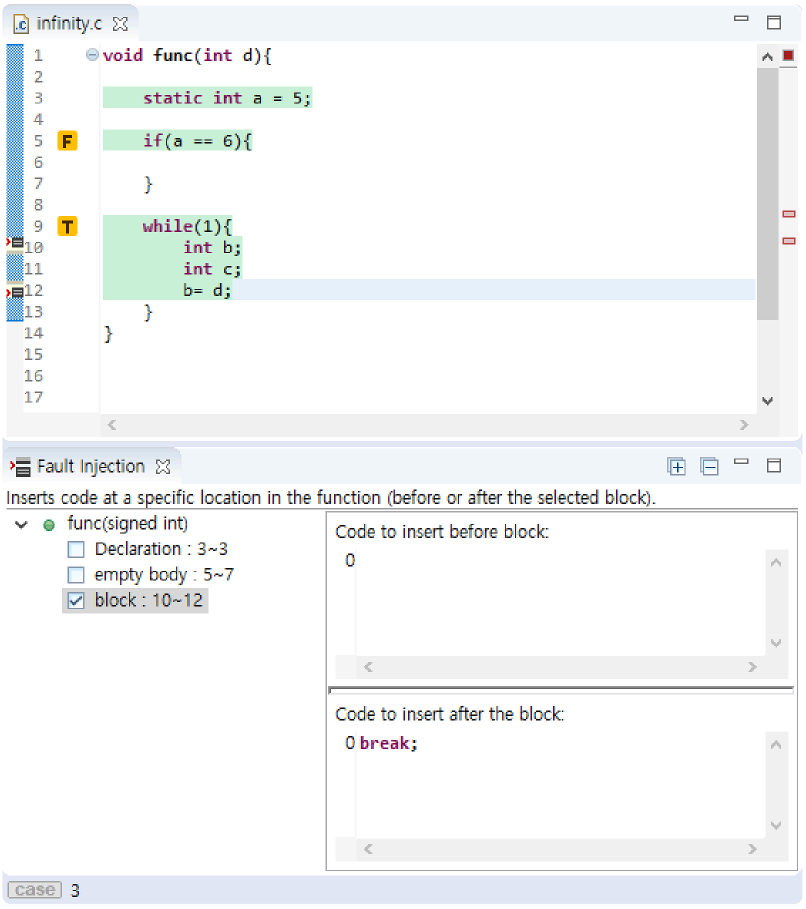
Code to insert before block:

```
0
```

Code to insert after the block:

```
0 b = 5;
```

- When a conditional statement that can break out of the infinite loop isn’t inside the loop.
  - Inject the fault at the end of the loop so that break out of the infinite loop.



```
void func(int d){

    static int a = 5;

    if(a == 6){

    }

    while(1){
        int b;
        int c;
        b= d;
    }
}
```

Inserts code at a specific location in the function (before or after the selected block).

- func(signed int)
  - ☐ Declaration : 3~3
  - ☐ empty body : 5~7
  - ☑ block : 10~12

Code to insert before block:

```
0
```

Code to insert after the block:

```
0 break;
```

1.47. If the file name is truncated when exporting tests

When exporting tests, if the file name is longer than the test code file name length limit, the file name may be truncated.
You can try the following methods to solve the problem.

1. Exit Controller Tester.
2. In the `workspace\project\.csdata\ut.ini` file, set the TEST_FUNCTION_NAME_LENGTH value to around 100, and then
Export by creating a new truncated test.
3. Run Controller Tester.

! When exporting, the export location + file name must be set shorter than the maximum path length allowed by Windows.

1.48. When an error ‘invalid use of void expression’ occurs in the host test using the conversion toolchain

The ‘invalid use of void expression’ error occurs when you try to assign a call to a void function to a variable. It also occurs when a void function operands with a different data type.

Even if the error does not occur in the original source file, the error may occur during the Controller Tester test process.

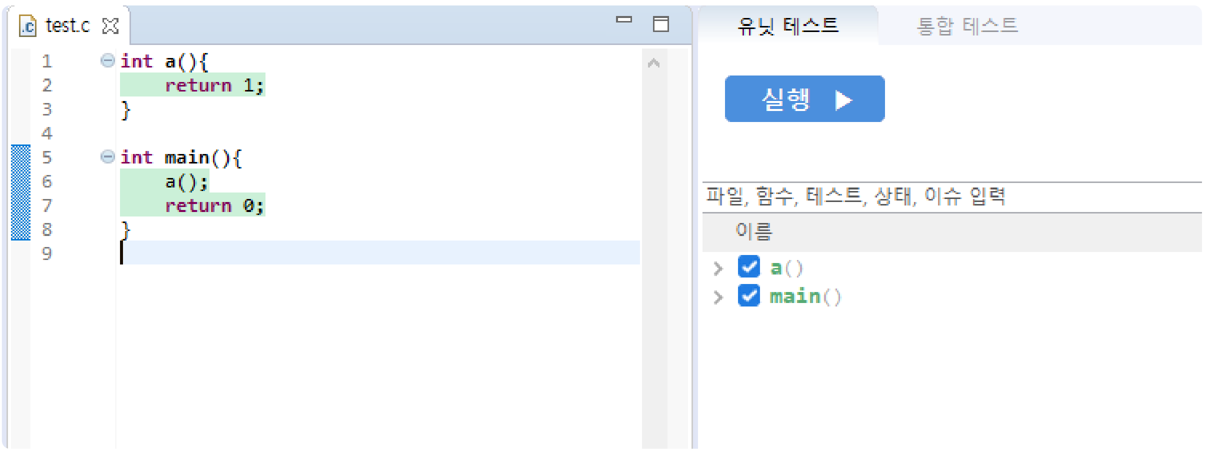
```
typedef unsigned int uint8_t;

typedef volatile uint8_t register8_t;

typedef struct PORT_struct
{
    register8_t DIR;
    ...
} PORT_t;

static static void PORTE_set_port_dir(void)
{
        uint8_t i;
        *((uint8_t *)&(*(PORT_t *) 0x0480) + 0x10 + i) |= 1 << 3;
}
```

The above example is using type punning and directly accessing the memory address by casting the constant to a pointer.
In this case, if the virtual memory option is being used, the contents of the function are wrapped with the void *CS_UT_get_host_addr(unsigned int addr, unsigned int size); function during conversion.
Since the CS_UT_get_host_addr function is of void type, the corresponding error occurs when operands as integers.

✻ If you add the add_cast_on_memorymap_operation = 1 option in the toolchain.ini file of %appdata%\Roaming\CodeScroll\1.1\parserConfig, the CS_UT_get_host_addr function is cast as the operand type during conversion.

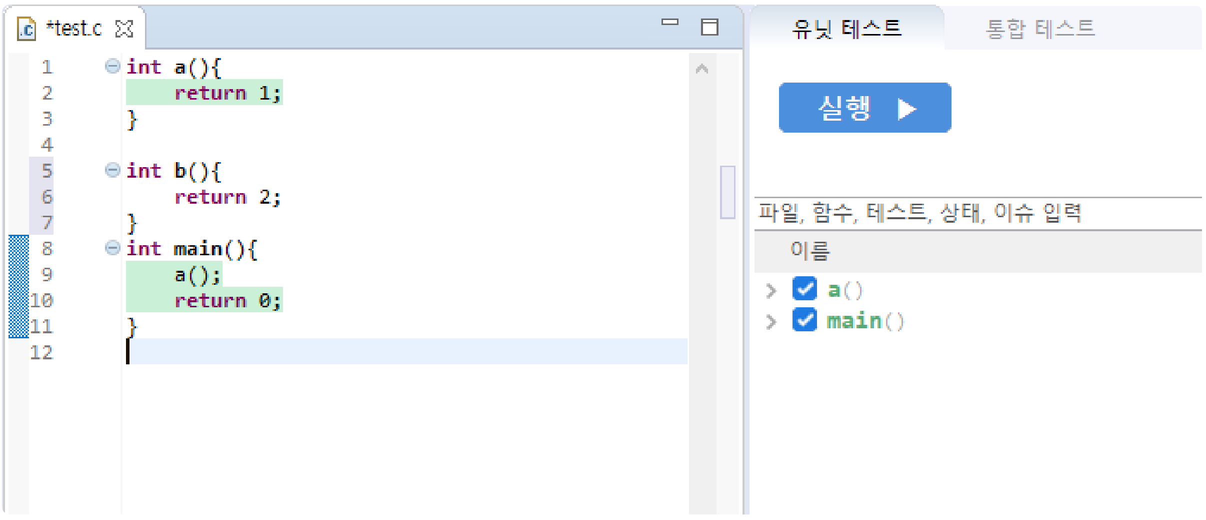
1.49. When tests need to be created only for new functions in a recycling scenario

You can only create tests for new functions in the following way.



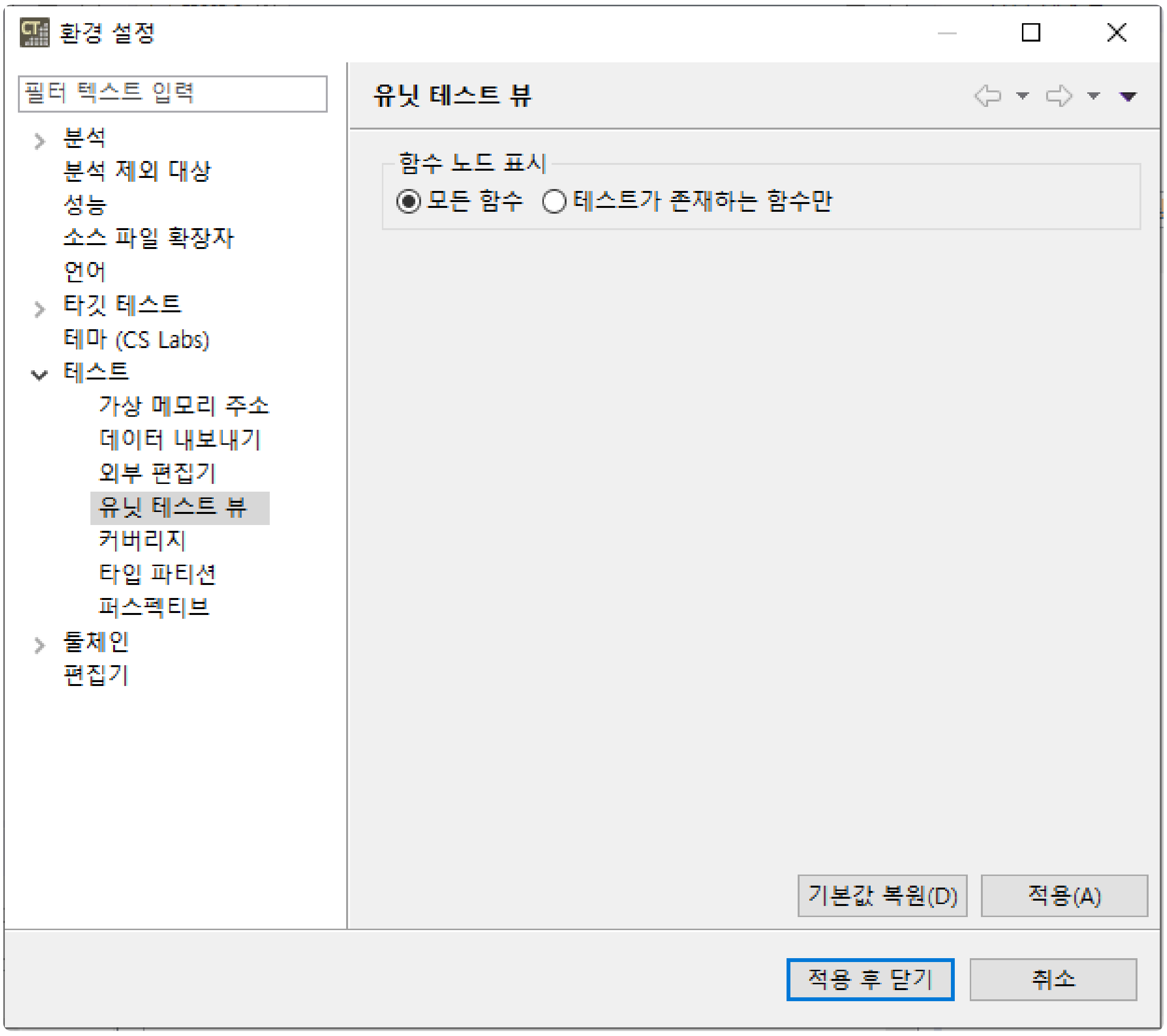
```
int a(){
    return 1;
}

int main(){
    a();
    return 0;
}
```

1. In this example, the tests for a and main functions have been completed.



```
int a(){
    return 1;
}

int b(){
    return 2;
}
int main(){
    a();
    return 0;
}
```

2. A new function b has been added to the source.

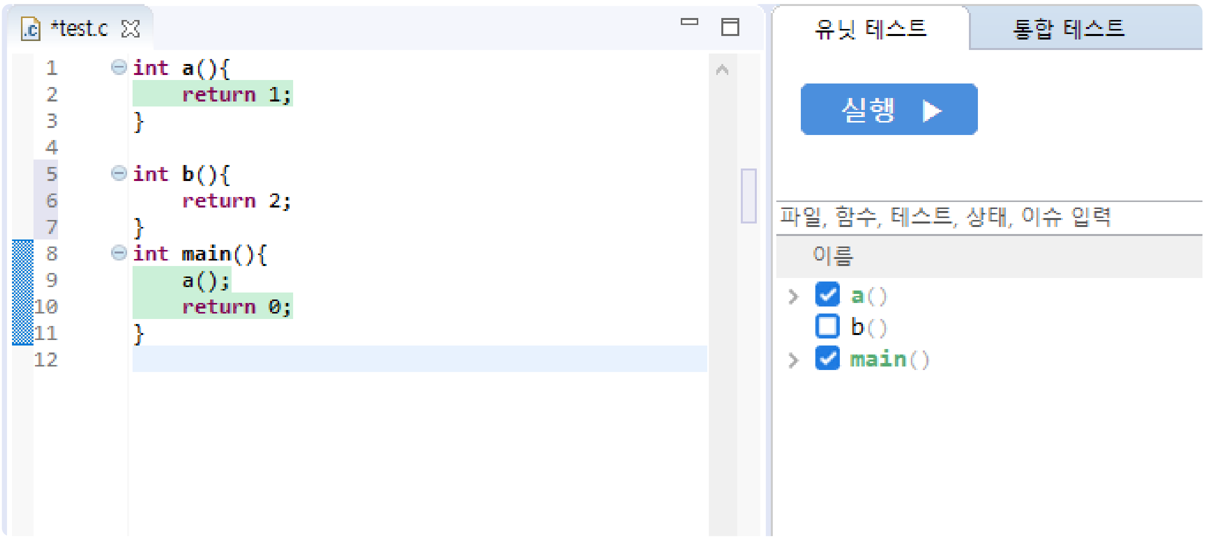


유닛 테스트 뷰

함수 노드 표시

[✓ 모든 함수] 테스트가 존재하는 함수만

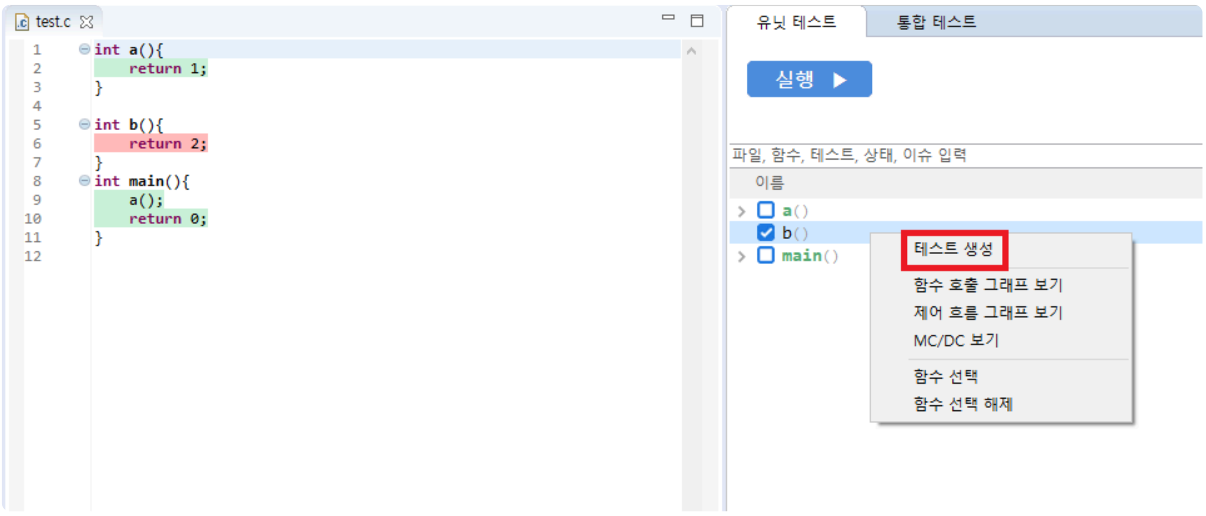
3. On the [Preferences] – [Tests] – [Unit Test View] page, select the option to show function nodes to [All Functions].



```
int a(){
    return 1;
}

int b(){
    return 2;
}
int main(){
    a();
    return 0;
}
```

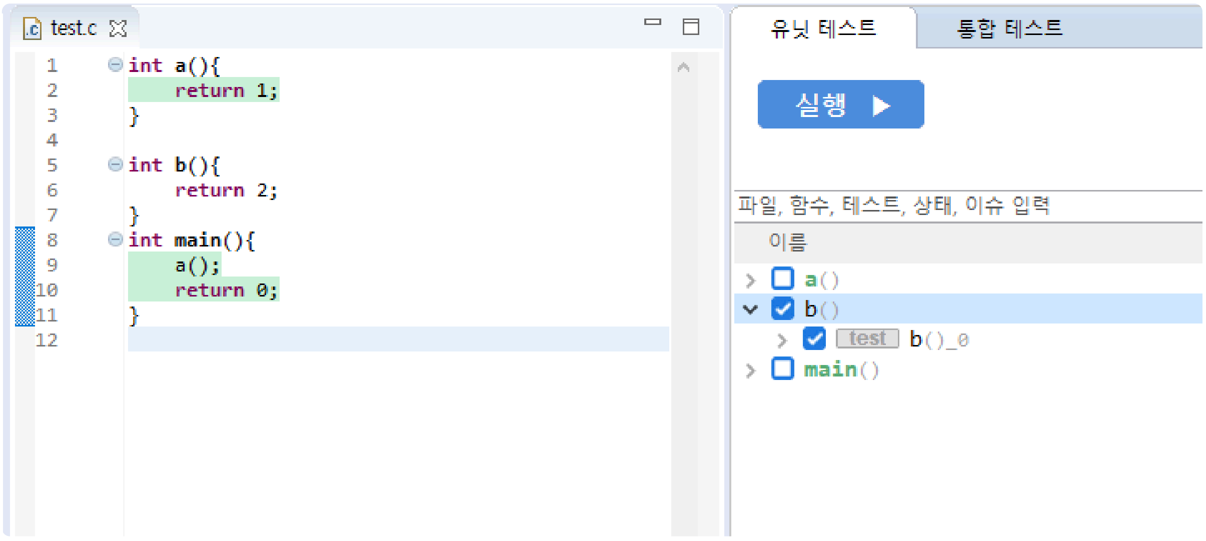
[✓] a()
[ ] b()
[✓] main()

4. Functions that have not created tests are visible in the unit tests view.



```
int a(){
    return 1;
}

int b(){
    return 2;
}
int main(){
    a();
    return 0;
}
```

5. In the unit test view, select only unchecked items (items without subtests) by pressing Ctrl + click, and then press [Create Test] in the context menu.

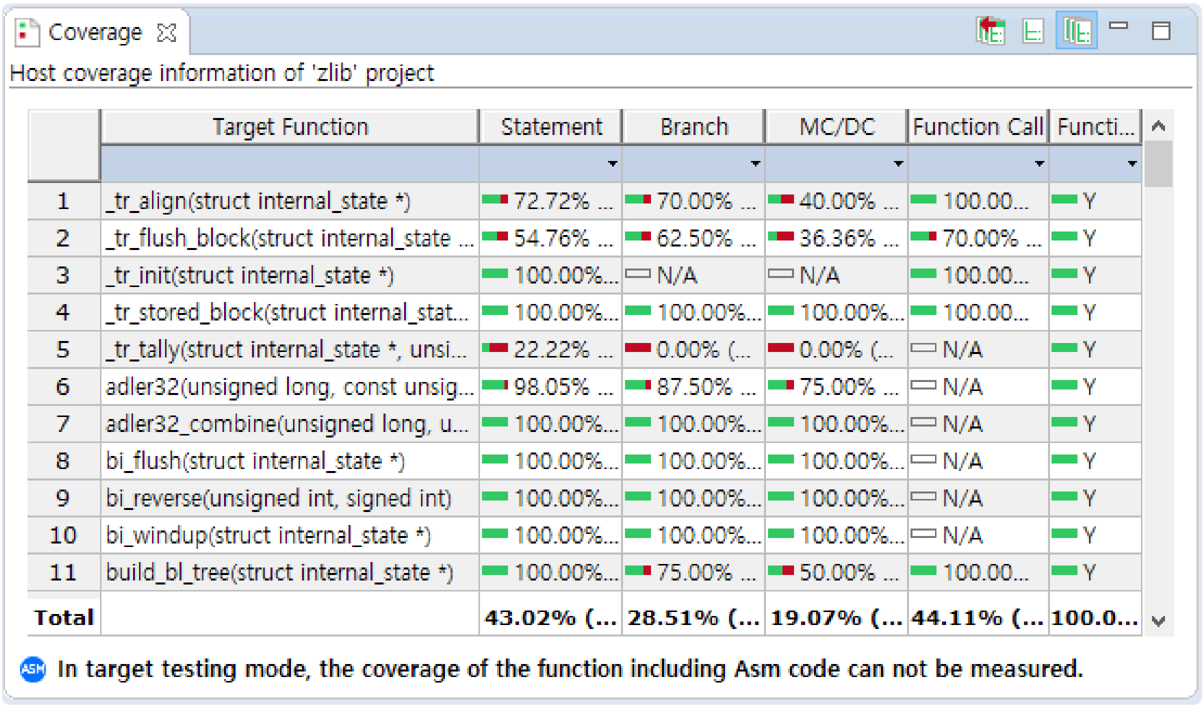


```
int a(){
    return 1;
}

int b(){
    return 2;
}
int main(){
    a();
    return 0;
}
```

6. By doing the above, you can create tests for functions newly added to the source.

1.50. If the color of the covered area does not change when selecting a test case

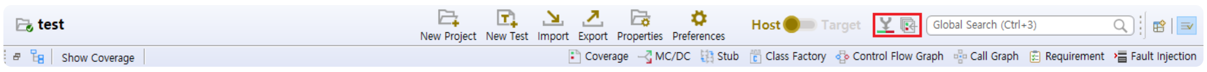
Coverage by test or test case is not displayed when the Show Coverage option is in [Show Full Coverage View (including External Coverage)].
Please turn off the [Show Full Coverage View (including External Coverage)] and check the coverage by test or test case.



Host coverage information of 'zlib' project

	Target Function	Statement	Branch	MC/DC	Function Call	Functi...
1	_tr_align(struct internal_state *)	72.72% ...	70.00% ...	40.00% ...	100.00...	Y
2	_tr_flush_block(struct internal_state ...	54.76% ...	62.50% ...	36.36% ...	70.00% ...	Y
3	_tr_init(struct internal_state *)	100.00%...	N/A	N/A	100.00...	Y
4	_tr_stored_block(struct internal_stat...	100.00%...	100.00%...	100.00%...	100.00...	Y
5	_tr_tally(struct internal_state *, unsi...	22.22% ...	0.00% (...	0.00% (...	N/A	Y
6	adler32(unsigned long, const unsig...	98.05% ...	87.50% ...	75.00% ...	N/A	Y
7	adler32_combine(unsigned long, u...	100.00%...	100.00%...	100.00%...	N/A	Y
8	bi_flush(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
9	bi_reverse(unsigned int, signed int)	100.00%...	100.00%...	100.00%...	N/A	Y
10	bi_windup(struct internal_state *)	100.00%...	100.00%...	100.00%...	N/A	Y
11	build_bl_tree(struct internal_state *)	100.00%...	75.00% ...	50.00% ...	100.00...	Y
Total		**43.02% (...**	**28.51% (...**	**19.07% (...**	**44.11% (...**	**100.0...**

In target testing mode, the coverage of the function including Asm code can not be measured.



1.51. Problem that the project does not open when changing the version from CT2.9 to CT3.0

You can try the two methods below to solve the problem.

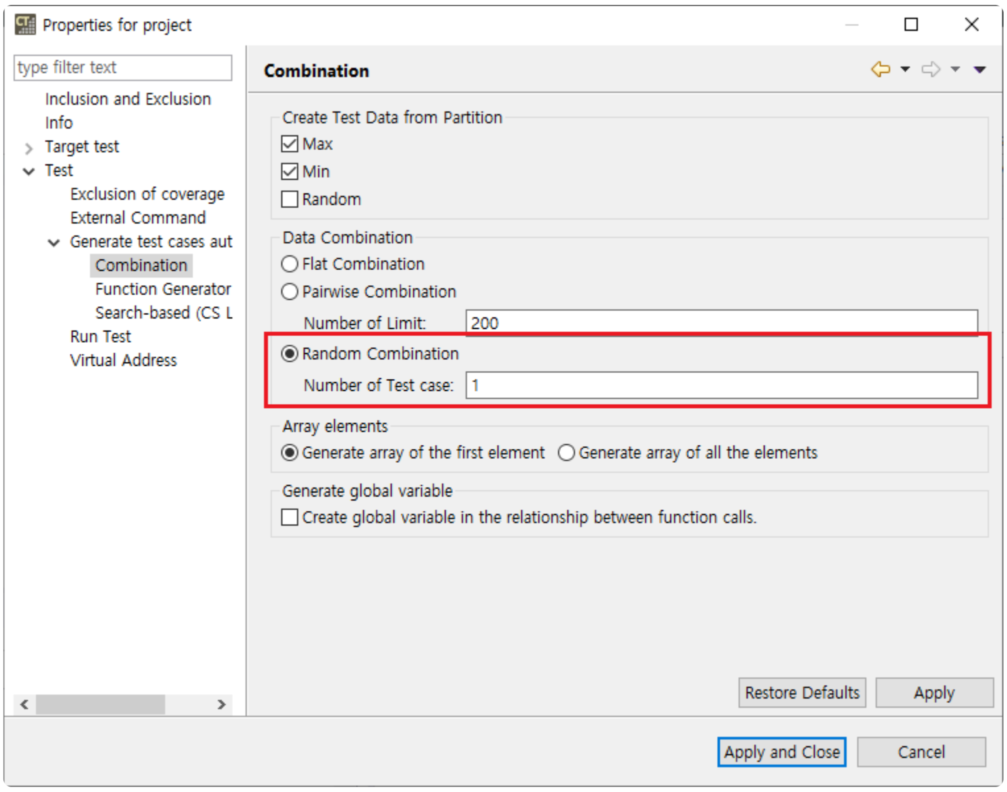
- Import project from Controller Tester
  1. Create a new workspace
  2. Import a project using the [Import] – [General] – [Existing Projects into Workspace] function in CT 3.0 version.

- Copy project from file system
  1. In the new workspace, create an empty project with the same name as the existing project
  2. Copy the project directory of the existing workspace from the file system as is
  3. After closing the project in Controller Tester, open it again

If the above does not solve the problem, there is a possibility that there is a problem with the installed 3.0 package shape. Please try again after reinstalling the package.

1.52. How to set only one test case to be created when creating a test

You can set the number of test cases to generate only one test case by changing the Data Combination option in [Project Properties] > [Test] > [Generate test cases automatically] > [Combination].



Combination

Create Test Data from Partition
- [✓] Max
- [✓] Min
- [ ] Random

Data Combination
- ○ Flat Combination
- ○ Pairwise Combination
  - Number of Limit: 200
- ● Random Combination
  - Number of Test case: 1

Array elements
- ● Generate array of the first element ○ Generate array of all the elements

Generate global variable
- [ ] Create global variable in the relationship between function calls.

1.53. Integrated coverage is displayed as 0 or the execution result is not visible after the test run

In the case of a project in which the virtual address is set in [Project Properties] > [Test] > [Virtual Address], if the range of the virtual address exceeds the available system memory range of the test execution environment (ex. 0×0-0×60000400), the following problems may occur.

- The integrated coverage of the test where the coverage result was normally displayed is displayed as 0
- Test execution results are not displayed after execution

Try to execute the test again after reducing the range of the set virtual address.

1.54. (After Ver.3.6) When you want to import test data as ‘overwrite’

After Controller Tester 3.5, the method of importing test data is set to ‘Append’, so existing data is not overwritten.
Since Controller Tester 3.6, you can set the method of imporing test data. The default is ‘Append’, and below is how to set the import method to ‘Overwrite’.

1. Open the `{Product installation path}\plugins\com.codescroll.gp.rcp.helios_xx\plugin_customization.ini` file.
2. Save the com.codescroll.ut/import.testdata.append option to false.

When executing Controller Tester, you can import test data as ‘overwrite’.

2. Controller Tester Target Plug-in Troubleshooting Guide

- Build issues after exporting target test code
- TRACE32 related issues
- Importing target log (test results) issues
- Other tips

2.1. Build issues after exporting target test code

- When the entry point function name is not main
- Multiple definition error of the function used to save the target test result
- ‘sprintf’ has not been declared or CS_FLT_OUTPUT error
- Target log interface settings
- Check for signal errors
- Problems When Testing Functions Related to UART Communication in Target Software
- When building with CodeWarrior, test related files such as cs_tfx.c are not found
- Target output value does not appear or is output as 0 for float type in CodeWarrior
- undefined error of type codescroll_int32 and codescroll_uint32
- When cannot use ‘long’ or undefined type to ‘long’ error occurs in codescroll_int, codescroll_uint type
- When the address of cs_io_putbyte is not found in Code Composer Studio
- When declaration is incompatible with “void cs_io_putbyte” error occurs in GreenHills AdaMulti

2.1.1. When the entry point function name is not main (for versions before CT 3.2)

The Controller Tester Target Plugin replaces the original main function with the cs_renamed_main function so that when the software is run on the target, the main function defined in Controller Tester is executed.
If the original entry point function name is not main, you need to modify the cs_tfx.c and cs_build_macro.h files.
The cs_tfx.c file is included in the Controller Tester installation path, but cs_build_macro.h is generated every time you export the target test code, so you need to apply the patch to fix the problem.

If you suspect an entry point function name problem, you can check the code below to modify it.

cs_tfx.c

```
#if defined main /* normal */
#undef main // replace with entry point function name
#endif
```

cs_build_macro.h

```
#if !defined CS_START_FROM_IUT
#define main        __cs_renamed_main // replace with entry point function nam
e
```

2.1.2. Multiple definition error of the function used to save the target test result (for versions before CT 3.2)

Target test results are output in the form of ‘target log’ when the test is run on the target. Because the target log uses the memset function to store data, it is declared with the extern keyword in the cs_tfx.h file to use the memset function.
When you export the target test code, the cs_tfx.h file is included in the original source code. At this time, multiple definition error may occur depending on whether memset or strncpy function is included in the source code to be tested.

```
ex) multiple definition of `memset' error
```

If you encounter this error, comment out the multiple error line in the cstfx.h file in the Controller Tester installation path and then ‘Export Target Test Code’ to fix the problem.

```
Example cstfx.h path: C:\Program Files\CodeScroll Controller Tester 3.1\plugin
s\com.codescroll.ut_3.1.2\target\lib\controller_nc\cs_tfx.h
```

2.1.3. ‘sprintf’ has not been declared or CS_FLT_OUTPUT error (for versions before CT 3.2)

Sometimes the sprintf function is not available in the source code under test.
In this case, you can fix the problem by adding the following code to the cstfx.h file in the Controller Tester installation path.

```
Example cstfx.h path: C:\Program Files\CodeScroll Controller Tester 3.1\plugin
s\com.codescroll.ut_3.1.2\target\lib\controller_nc\cs_tfx.h
```

Add

```
int sprintf(char* str, const char* format, ...);
```

to line 210 of the cs_tfx.h file

2.1.4. Target log interface settings

The target log interface file name is cs_io_implementation.c. Performing ‘Export Target Test Code’ will include the target log interface file for each source file.
Using the target log interface’s cs_io_putbyte function, the log is output in char units. Depending on the implementation of the target log interface, the target log can be output in various ways such as Ethernet, UART (Serial), and JTAG.
Depending on the communication method used, appropriate initialization logic must be implemented in the cs_io_initialize function. In particular, if you do not set the baudRate or dataBits settings when using UART, the target log may be displayed incorrectly.
Accurate setup for target testing requires an understanding of the target and software, so you should contact your target development staff.

2.1.5. Check for signal errors

You can check the output of the target log to see if an error occurred while the test was running in the target environment.
You can check the target log to see if an error occurred during test case execution or if an error occurred before the test case was run.
When the test run is completed normally, “CSET” is printed at the end of the target log. If the log is truncated after “CSTC” is printed, you should find the test case in the project DB.

```
Project DB file: { ProjectName }.csp file in .csdata directory under project p
ath in [Project Properties]-[Info]-[Location]
Testcase output format: CSTC <TEST_ID, TEST_CASE_NO>
```

Target log output example

```
CSTR<374069146,130789013,1,0,1>CSTR#
CSST<test,1574932569>CSST#
CSTC<133143986176,1>CSTC#
CSOS<1returnVar,0>CSOS#
CSES<1>CSES#
CSTB<1>CSTB#
CSET< test,1574932569>CSET#
```

2.1.6. Problems When Testing Functions Related to UART Communication in Target Software

If a test case is performed that changes the UART communication settings, subsequent target logs may not be output correctly.
(Example: test case to change baudrate setting)
In this case, you must switch from the settings in the test case editor to unmanaged code, and then modify (implement the code) the test case to back up the settings before the test case is executed and restore them after the test case is run.

2.1.7. When building with CodeWarrior, test related files such as cs_tfx.c are not found

In CodeWarrior's project settings, make sure the system path contains the following path, and add it if it doesn't exist.

```
- {CT_Workspace}/.metatdata/.plugins/com.codescroll.ut.embedded/{ ProjectName
}/TestFixture/cs
- {CT_Workspace}/.metatdata/.plugins/com.codescroll.ut.embedded/{ ProjectName
}/TestFixture
```

2.1.8. Target output value does not appear or is output as 0 for float type in CodeWarrior

When the variable value of the real type is output in the target test, the result value is retrieved through the sprintf function.
However, there are cases where the library provided by CodeWarrior cannot use the sprintf function for real types.
In this case, you need to replace or add the library linked from the CodeWarrior project.
Libc99_E200z650.a is a library that can use the sprintf function for the real type currently identified.
After replacing with the library, edit the contents of the cs_tfx.h file as follows.

Before modification

```
#define TFX_ftoa_writeBytes(value) \
do {\
        sprintf(buf, "%g", value);\
        TFX_writeBytes(buf);\
} while(0)
```

After modification

```
 #define TFX_ftoa_writeBytes(value) \
do {\
        sprintf(buf, "%f", value);\
        TFX_writeBytes(buf);\
} while(0)
```

2.1.9. undefined error of type codescroll_int32 and codescroll_uint32

Check if there are codescroll_int32 and codescroll_uint32 in the info file of the toolchain set in the project. If not, add the type. After editing the info file, you will need to ‘Export Target Test Code’ again.

```
Toolchain info file path: { ToolchainName }.info in [Preferences]-[ToolChain]-
[Open configuration Folder]
```

2.1.10. When cannot use ‘long’ or undefined type to ‘long’ error occurs in codescroll_int, codescroll_uint type

This error can occur if the target software does not support the long type. Change the codescroll_int and codescroll_uint type to int in the info file of the toolchain set in the project.

```
Toolchain info file path: { ToolchainName }.info in [Preferences]-[ToolChain]-
[Open configuration Folder]
```

```
ex) unsigned long long,unsigned,0,9223372036854775807,8,codescroll_uint
        -> unsigned int,unsigned,0,9223372036854775807,8,codescroll_uint
```

2.1.11. When the address of cs_io_putbyte is not found in Code Composer Studio

```
SEVERE: Unable to get address for symbol: cs_io_putbyte
org.mozilla.javascript.WrappedException: Wrapped com.ti.ccstudio.scripting.env
ironment.ScriptingException: Unable to get address for symbol: cs_io_putbyte
```

If the target_binary_path in the execution tab of the target environment setting and the actually built binary are different, the error may occur.

In this case, it works normally if you change the target_binary_path to the path of the actual built binary.

The way to find out in the console log that this is the cause of the error is as follows.

```
Check if the .out file of the finished building target and the .out file of lo
adProgram: ENTRY sFileName: are the same
```

2.1.12. When declaration is incompatible with “void cs_io_putbyte” error occurs in GreenHills AdaMulti

If **declaration is incompatible with “void cs_io_putbyte(code scroll_byte)”** error occur when build in AdaMulti, it is because `codescroll_byte` type is set as `signed char` or `unsigned char`, not `char`. Modify `codescroll_type` type in .info file of toolchain used in project analysis as the below. **(You can find the .info file of toolchain in ‘Preferences’ > ‘ToolChain’ > ‘Open Configuration Folder’)**

Before modification

```
#typeName,valueKind,min,max,size,csType
signed char,signed,-128,127,1,codescroll_byte
```

After modification

```
#typeName,valueKind,min,max,size,csType
char,signed,-128,127,1,codescroll_byte
```

After modification, select the toolchain and click ‘edit’ > ‘Finish’ > ‘Apply and Close’ in ‘Preferences’ > ‘ToolChain’

2.2. TRACE32 related issues

- symbol not found error “ct_target_log”
- target reset failed
- ‘&binary_path’ symbol not found or an error occurred at Data.Load.Elf “{file_path}” /LPATH while running the cmm script

2.2.1. symbol not found error “ct_target_log”

The results of the tests on the target are stored in ct_target_log.
If you get an error that can’t find the ct_target_log, there are two things to check:

1. Make sure your target test code is built correctly
    - If the test code was not applied correctly or if the binaries built with the test code were not deployed to the target (build the original software only), you might get an error that a variable cannot be found to store the test results.

2. Position of the GO.HLL instruction in the cmm script
    - You must write a cmm script to specify a breakpoint with the BREAK.SET command and begin debugging.

ex)

```
BREAK.SET cs_io_initialize /Program /cmd " … " /RESUME
BREAK.SET cs_write_log /Program /cmd "…"  /RESUME
BREAK.SET cs_io_finalize /Program /cmd "…" /RESUME /cmd "…" /RESUME

GO.HLL <- Enter after setting BREAK
```

2.2.2. target reset failed

There are two things to check.

1. Make sure the CPU_NAME in System.CPU {CPU_NAME} is set correctly in the cmm script
2. Make sure that the target binary size is too big to load on the target

2.2.3. ‘&binary_path’ symbol not found or an error occurred at Data.Load.Elf “{file_path}” /LPATH while running the cmm script

When testing targets using Trace32, the Controller Tester calls the target.cmm script from the start.cmm script.
When calling the target.cmm script from the start.cmm script, pass “&binary_path” as a parameter. This error can occur if the build is not successful or if the path passed as a parameter is incorrect.
In this case, you should check that the build is successful and check the path in the start.cmm script.

2.3. Importing target log (test results) issues

- Failed or Error When Performing ‘Import Target Test Log’
- When the target log buffer size is exceeded
- When an exception occurs in the script when running after test building in Code Composer Studio
- Fail to import debug information logs in target test
- How to read target logs when fail ‘Import Target Test Log’

2.3.1. Failed or Error When Performing ‘Import Target Test Log’

You should open and check the target log file. If the log is output normally, the log ‘CSET <…> # CSET’ is output last. If this log is not printed, a signal error likely occurred while running tests on the target.
To fix the problem, run the test cases one by one and find and fix the test case that caused the signal error.

2.3.2. When the target log buffer size is exceeded

If you store the char value of the cs_io_putbyte function in a fixed-size buffer, such as char * buff[BUFF_SIZE], large test results can cause you to exceed the buffer size.
To fix the problem, modify the BUFF_SIZE in the “Target Log Interface” and export the target test code again.

2.3.3. When an exception occurs in the script when running after test building in Code Composer Studio

When an error occurs:

*CCS Caused by: com.ti.ccstudio.scripting.environment.ScriptingException: Could not open session. Found devices matching: .**

When running the target test using the Code Composer Studio debugger in Controller Tester, Controller Tester uses the execution script written in Javascript.
The script opens a debugging session from the available debugging probes and runs the target binary.
Basically, there is one debugging probe and one debugging session. Depending on the target environment setting, there may be two or more debugging probes, and the above error occurs.
You can fix the error by specifying the debugging probe you want to use.
The debugging probe can be set in Project Properties > Target Test > Target environment settings > [Run] tab > **debug_probe** option.

When you do not know which debugging probe to choose, run the target project in debugging mode in the Code Composer Studio IDE and set the debugging probe value displayed in the debug view to the debug_probe option.

When an error occurs: invalid target memory page

When you build the Code Composer Studio project, a .map file is generated. The .map file contains information about the address and page location to which variables are assigned.
The Code Composer Studio execution script provided by Controller Tester is implemented to store the target test result value in the data area and get this value.
However, if it is stored in the program area, not in the data area, an error message may appear when the script is executed, indicating that the memory area cannot be accessed.
Check the **char_format** option in Project Properties> Target Test > Target environment settings > [Run] tab.

2.3.4. Fail to import debug information logs in target test

You can check debug informations using logs when using ‘Inspect Debug Info’ feature during target tests without debugger. Debug information is displayed instead of coverage measurement when export test codes to target environment and build and run the test by using ‘Inspect Debug Info’. At this time, the default log output size is set to 1000. If the log information exceeds 1000 characters, execution will be terminated as it is, and the debugging log may be truncated. You can adjust the log output size through the **TARGET_DEBUG_PROBE_INDEX** option in the ut.ini file created for each project.

location of ut.ini file : %Controller Tester_project_path%\.csdata\ut.ini
option : TARGET_DEBUG_PROBE_INDEX=1000

2.3.5. How to read target logs when fail ‘Import Target Test Log’

Results of test execution is displayed as the below.

```
CSTR<212668380,3136998082,2,0>CSTR#
CSST<ct_test_log,425747>CSST#
CSTC<4294967296,2>CSTC#
CSOS<2s_aDoJobCnt[0],0>CSOS#
CSES<1>CSES#
CSTB<11001100000010010110000000>CSTB#
CSET<ct_test_log,425747>CSET#
```

- When an error occurs for execution of test case #2.

```
CSTR<212668380,3136998082,2,0>CSTR#
CSST<ct_test_log,425747>CSST#
CSTC<4294967296,2>CSTC# // Second number between CSTC tag means test case numb
er.
```

- When a specific variable in the test case is null and the output value is checked.

```
CSTR<212668380,3136998082,2,0>CSTR#
CSST<ct_test_log,425747>CSST#
CSTC<4294967296,2>CSTC#
CSOS<2s_aDoJobCnt[0],0>CSOS# // An error occurs during printing a value of s_a
DoJobCnt[0] because the value is null.
```

- When do not print coverage results because of infinite loop during test execution.

```
CSTR<212668380,3136998082,2,0>CSTR#
CSST<ct_test_log,425747>CSST#
CSTC<4294967296,2>CSTC#
CSOS<2s_aDoJobCnt[0],0CSOS#
CSES<1>CSES#
CSTB<11001100000010010110 // Coverage bit map is printed no longer because of
infinite loop during printing.
```

2.4. Other tips

- .map file
- TRACE32 debugging
- If the output value is different due to the byte order of the host/target environment

2.4.1. .map file

After building the target software, a .map file may also be created in the path where the binaries are created.
The file contains symbol information for the target binary. You can check whether the main function has been replaced normally, or check the function symbols contained in the binary file.

2.4.2. TRACE32 debugging

If you use TRACE32, you can debug your test binaries.

1. Run TRACE32 and open the cmm script file (start.cmm).
2. Run “Debug”, set a breakpoint after Go.HLL in the script, and run “Continue”.
3. When the break occurs, select [view]-[Var] to see the variable and function information currently running on the target.

2.4.3. If the output value is different due to the byte order of the host/target environment

If the byte order of the host environment and the target environment is different, the test fails because the host output value and the target output value are different.

When the output value of host/target is different

If the source code directly controls memory, the output value may vary depending on the byte order of the host environment and the target environment. Testing with the same expected values will cause either the host test or the target test to fail. In this case, you can run host/target tests with the same test case by using vertical bar(`|`) for the expected value.

- Example
    - Host output value : `0x1122`
    - Target output value : `0x2211`
    - Expected value to be entered : `0x1122|0x2211`

When the source code logic is completely dependent on byte order

In host test, logic that depends on how the memory is used will not have normal results. The following cases are dependent on how the memory is used.

- When using union to directly access and use memory details.
- Packing for bitfields that may have different implementations depending on the compiler.

The above type of code should be avoided in consideration of portability. Even when checking the test result, it is better to design to check the value of the data type rather than check the memory value directly.

If it is difficult to modify the source code, tests for such code should be performed in an actual target environment or host and target tests should be designed and managed separately.