Integrated Development Environment e² studio

How to use CUnit in e² studio (GCC for RX)

Introduction

CUnit is a system for writing, administering, and running unit tests in C. It is built as a library (static or dynamic) which is linked with the user's testing code.

CUnit uses a simple framework for building test structures and provides a rich set of assertions for testing common data types. In addition, several different interfaces are provided for running tests and reporting results.

This document describes how to use CUnit to automate unit testing using Renesas GCC Executable projects created in e² studio.

Contents

1. Overview

1.1 Purpose

This document describes how to use CUnit to automate unit testing using Renesas GCC Executable projects created in e² studio.

1.2 Operating Environment

Target device	RX610
IDE	e² studio 2021-07
Toolchains	GCC for Renesas RX C/C++ Toolchain v 8.3.0.202102
CUnit version	2.1.2

1.3 CUnit references

Further information about CUnit can be referred in http://cunit.sourceforge.net/doc/index.html.

2. Getting started with Cunit

This section shows how to setup CUnit to e² studio.

[Important notes]

- Download and use CUnit-2.1-2. CUnit--2.1-3 has some problems which causes build errors. Besides, CUnit-2.1-2 package lacks header file “ExampleTests.h”. Don’t build examples.
- The compiler (and Windows system) does not support “curse” module. Don’t build “curse”.

2.1 Building CUnit library

CUnit can be built to be a static library to be linked to user’s code. This section shows how to build the static library.

1) Download CUnit-2.1.2 from https://sourceforge.net/projects/cunit/files/CUnit/2.1-2/. Extract compressed file to get CUnit package.
2) Launch e² studio. In "C/C++" perspective, click [File] > [New] > [Renesas C/C++ Project] > [Renesas RX].
3) In the [Templates for New C/C++ Project] dialog, choose Renesas RX in the left-hand margin and "GCC for Renesas RX C/C++ Library Project" and click [Next >] button.

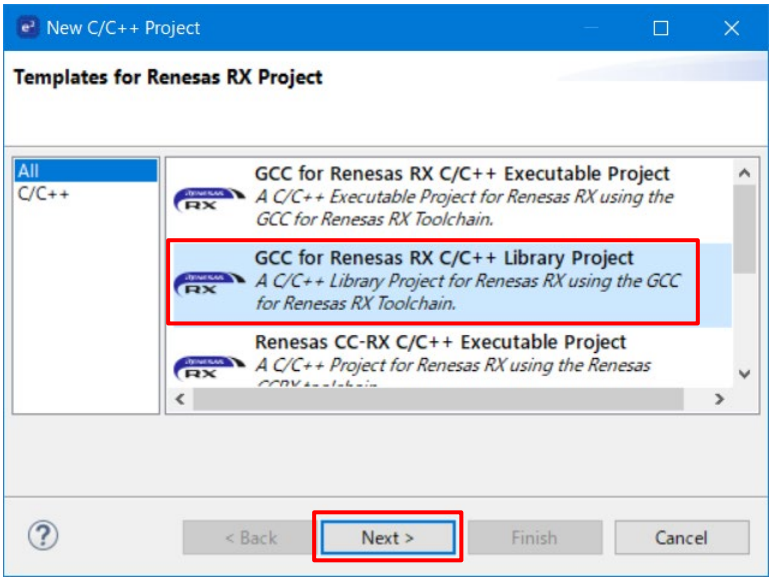

Figure 1 New C/C++ Library Project

4) In [Project name:] enter the name "CUnit" and click [Next >] button.
5) In the [Select toolchain, device & debug settings] page, enter the following information (other values can remain at default):
   - Toolchain: "GCC for Renesas RX"
   - Toolchain Version: "8.3.0.202102" or later version
   - Target Device: e.g.; "R5F56107VxFP"

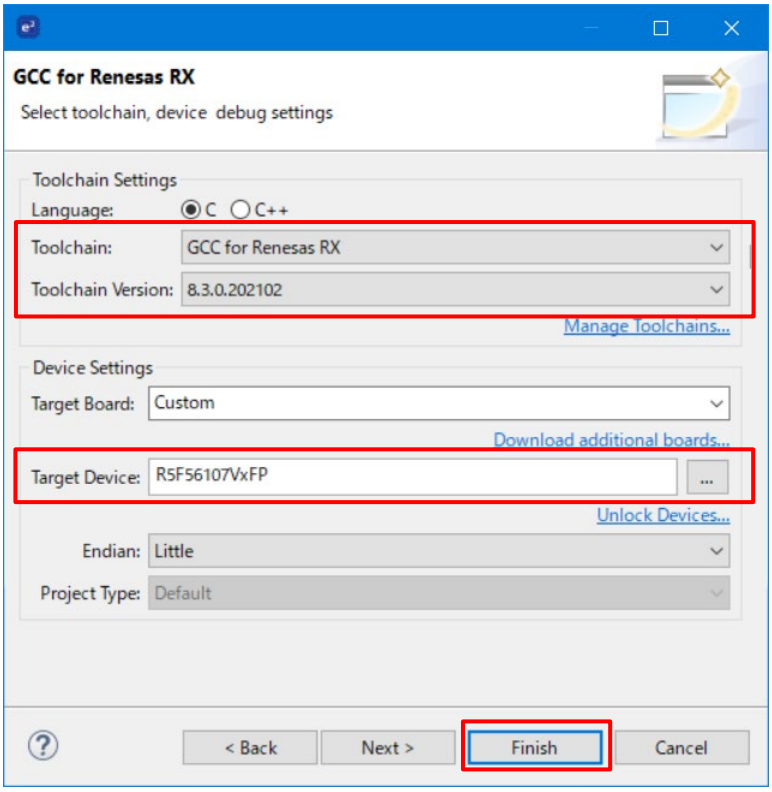

Figure 2 Toolchain and device settings

6) Click [Finish] button.

7) In the Project Explorer view, expand the CUnit project and delete files in the folder "src".

8) From the CUnit directory, downloaded and extracted previously, copy Headers and Sources subdirectories in CUnit into the "src" folder in CUnit library project. This can be accomplished, in Windows, using either the clipboard or by drag and drop from a File Explorer into e² studio.

9) In the CUnit library project "Sources" folder, delete the "Curses", "Test" and "Win" folders. Optionally, delete all files called "Makefile.*" from the "Sources" folder.

10) The project should resemble the figure below:

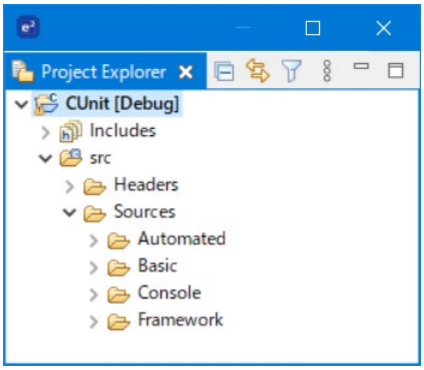

Figure 3 Copied files to e² studio project

11) Open project properties, select [C/C++ Build] > [Settings], [Compiler] > [Includes], then in [Include file directories (-I)] click [Add...] button and add include file directory "${workspace_loc:/${ProjName}/src/Headers}". Next click [Apply and Close] button.

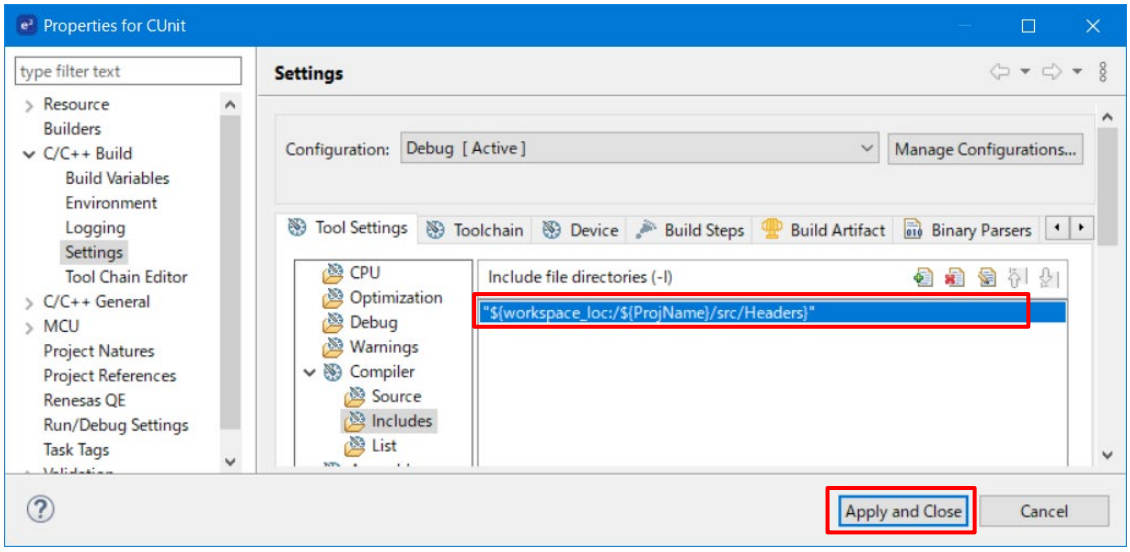

Figure 4 Add CUnit Header files directory to build setting

12) Build the project. The file “libCUnit.a” will appear inside the “Archives” folder, as shown in the figure below.

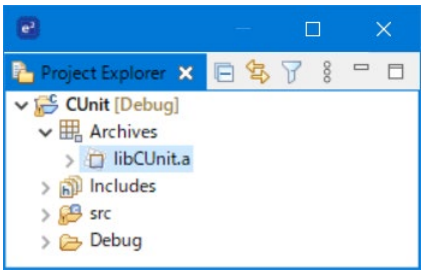

Figure 5 Output static library

The CUnit library file, “libCUnit.a”, can now be used in any C/C++ project to provide a CUnit test framework.

2.2 Performing unit testing using CUnit

1) In "C/C++" perspective, click [File] > [New] > [Renesas C/C++ Project] > [Renesas RX].

2) In the [Templates for New C/C++ Project] dialog, choose Renesas RX in the left-hand margin and "GCC for Renesas RX C/C++ Executable Project" and click [Next >] button.

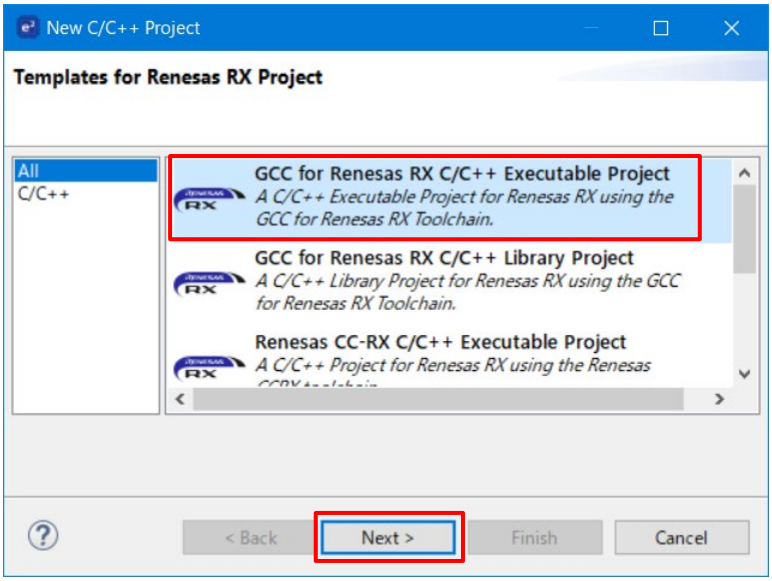

Figure 6 New Executable C/C++ project

3) In [Project name:] enter the name "SampleCUnit" and click [Next >] button.

4) In the [Select toolchain, device & debug settings] page, enter the following information (other values can remain at default):

- Toolchain: "GCC for Renesas RX"
- Toolchain Version: "8.3.0.202102"
- Target Device: e.g.; "R5F56107VxFP"
- Uncheck [Create Hardware Debug Configuration]
- Check [Create Debug Configuration] for “RX Simulator”.

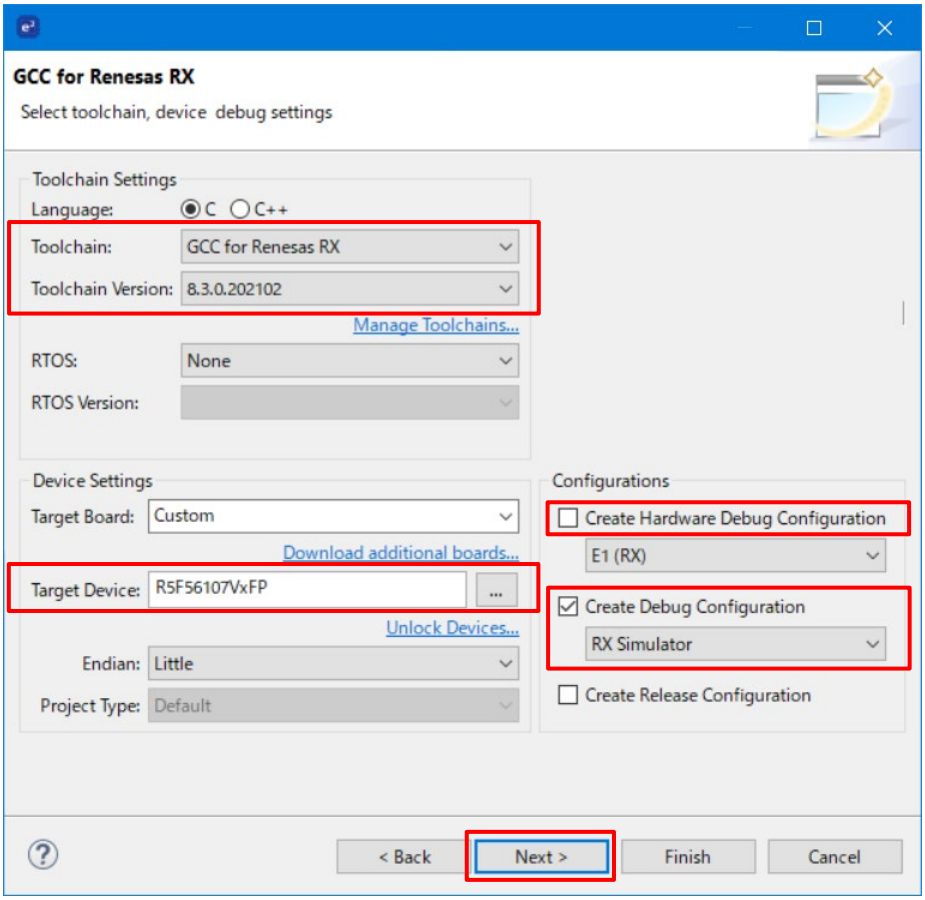

Figure 7 Toolchain and device settings

5) Keep clicking [Next >] button until the [Select library generator settings] page is reached. In [Select Library Source] choose "Newlib" and in [Select Library Type] choose the default "Project-Built". Click [Finish] button.

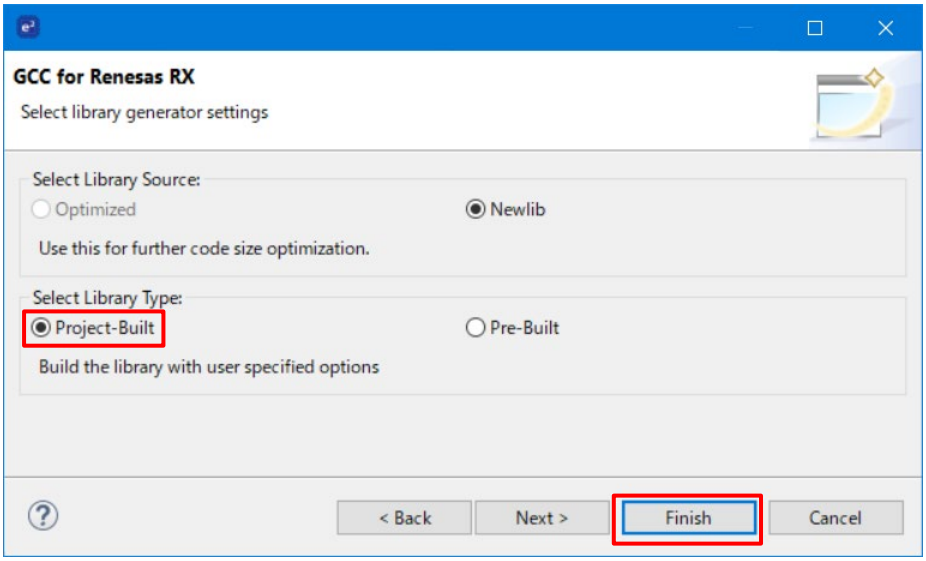

Figure 8 Select library generator settings

6) Create the following new files to be tested in “src” folder:

- source.h

```
#ifndef SOURCE_H_
#define SOURCE_H_
```

```
int add(int a, int b);
int subtract(int a, int b);

#endif
/* SOURCE_H_ */
```

- source.c

```
#include "source.h"

int add(int a, int b) {
  return a + b;
}

int subtract(int a, int b) {
  return a - b;
}
```

- testsource.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "CUnit.h"
#include "source.h"

// This is a test case used to test add() function in source.c
static void test_Add_01(void) {
  // Equal Assertion is used in this test case.
  // 1 is expected value, and add(1,0) is actual return value.
  // If expected value is not same, assertion occurs.
  // We can refer the Reference document for the other useful
assertion.
  CU_ASSERT_EQUAL(1, add(1,0));
}

static void test_Add_02(void) {
  CU_ASSERT_EQUAL(10, add(1,9));
}

// This is a test case used to test subtract() function in source.c
static void test_Subtract(void) {
  // 0 is expected value, and subtract(1,1) is actual return value.
  // If expected value is not same, assertion occurs.
  CU_ASSERT_EQUAL(0, subtract(1,1));
}

// This is a test suite
static CU_TestInfo tests_Add[] = {
  // Register test case to test suite
  {"test_Add_01", test_Add_01},
  {"test_Add_02", test_Add_02},
  CU_TEST_INFO_NULL,
};

static CU_TestInfo tests_Subtract[] = {
  {"test_Subtract", test_Subtract},
```

```
  CU_TEST_INFO_NULL,
};

// Declare the test suite in SuiteInfo
static CU_SuiteInfo suites[] = {
  {"TestSimpleAssert_AddSuite", NULL, NULL, tests_Add},
  {"TestSimpleAssert_SubtractSuite", NULL, NULL, tests_Subtract},
  CU_SUITE_INFO_NULL,
};

void AddTests(void) {
  // Retrieve a pointer to the current test registry
  assert(NULL != CU_get_registry());

  // Flag for whether a test run is in progress
  assert(!CU_is_test_running());

  // Register the suites in a single CU_SuiteInfo array
  if (CU_register_suites(suites) != CUE_SUCCESS) {
    // Get the error message
    printf("Suite registration failed - %s\n", CU_get_error_msg());
    exit(EXIT_FAILURE);
  }
}
```

7) Replace the contents of the existing source file, “SampleCUnit.c”, and add code to run the test

- SampleCUnit.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Basic.h"

int main(void);
extern void AddTests();

int main(void)
{
  // Define the run mode for the basic interface
  // Verbose mode - maximum output of run details
  CU_BasicRunMode mode = CU_BRM_VERBOSE;

  // Define error action
  // Runs should be continued when an error condition occurs (if
possible)
  CU_ErrorAction error_action = CUEA_IGNORE;

  // Initialize the framework test registry
  if (CU_initialize_registry()) {
    printf("Initialization of Test Registry failed.\n");
  }
  else {
    // Call add test function
    AddTests();

    // Set the basic run mode, which controls the output during test
runs
```

```
    CU_basic_set_mode(mode);

    // Set the error action
    CU_set_error_action(error_action);

    // Run all tests in all registered suites
    printf("Tests completed with return value %d.\n",
CU_basic_run_tests());

    // Clean up and release memory used by the framework
    CU_cleanup_registry();
  }
  return 0;
}
```

8) Create the following new files to be tested in “generate” folder:

- sbrk.c

```
void*
sbrk(int incr)
{
  extern char end; /* Set by linker. */
  static char * heap_end;
  char * prev_heap_end;

  if (heap_end == 0)
    heap_end = &end;

  prev_heap_end = heap_end;
  heap_end += incr;

  return (void *)prev_heap_end;
}
```

9) Edit the file generate/start.S and change the definition of _exit: so it is empty. This will allow the program to terminate at the end of the test run.

- Generate/start.S

```
:
:
/* call to exit*/
_exit:
  brk
/*
  mov #0, r2
  mov #___call_exitprocs, r7
  jsr r7
_loop_here:
  bra _loop_here
*/

  .text
  .end
```

10) Open project properties, select [C/C++ Build] > [Settings], [Compiler] → [Includes], then in [Include file directories (-I)] click [Add...] button and add the include file directory from the CUnit project, "${workspace_loc:/CUnit/src/Headers}".

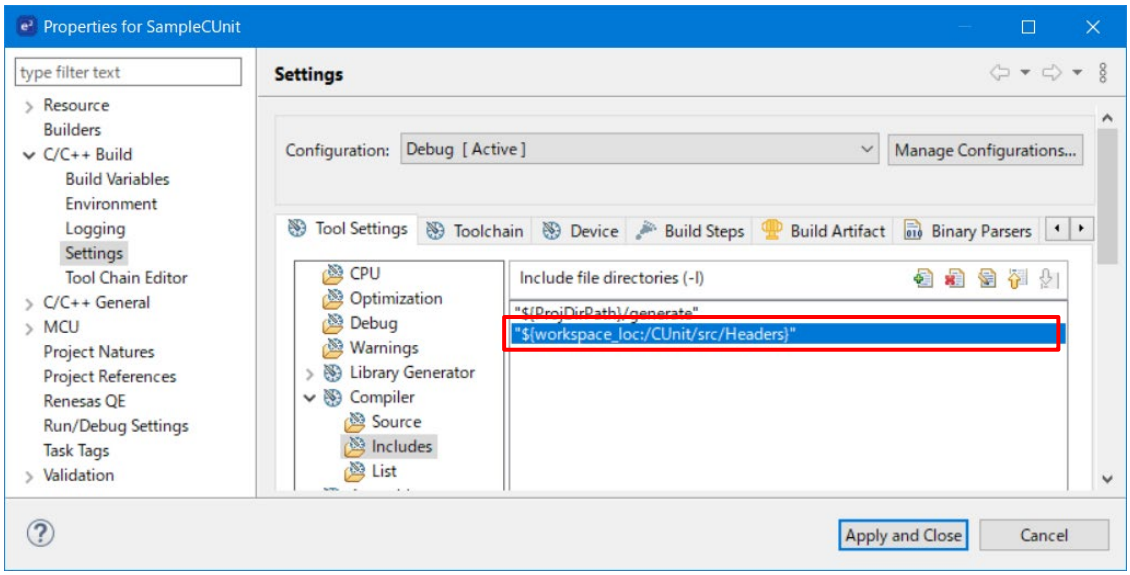

Figure 9 Add CUnit header files to build

11) In [Linker] > [Source], [Additional input files], add the CUnit library "${workspace_loc:/CUnit/Debug/libCUnit.a}".

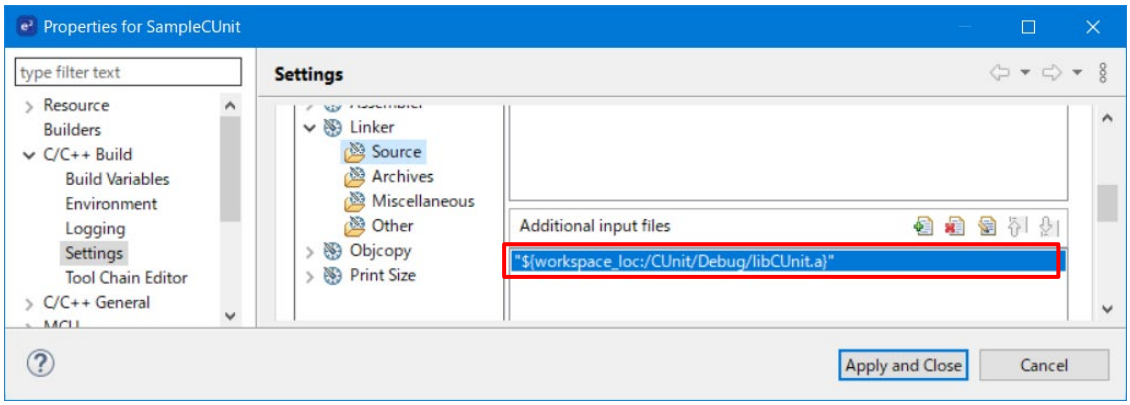

Figure 10 Add CUnit library to linker

12) In [Optimization], tick the [No common uninitialized (-fno-common)] checkbox, as in the figure below, and click [Apply and Close] button.

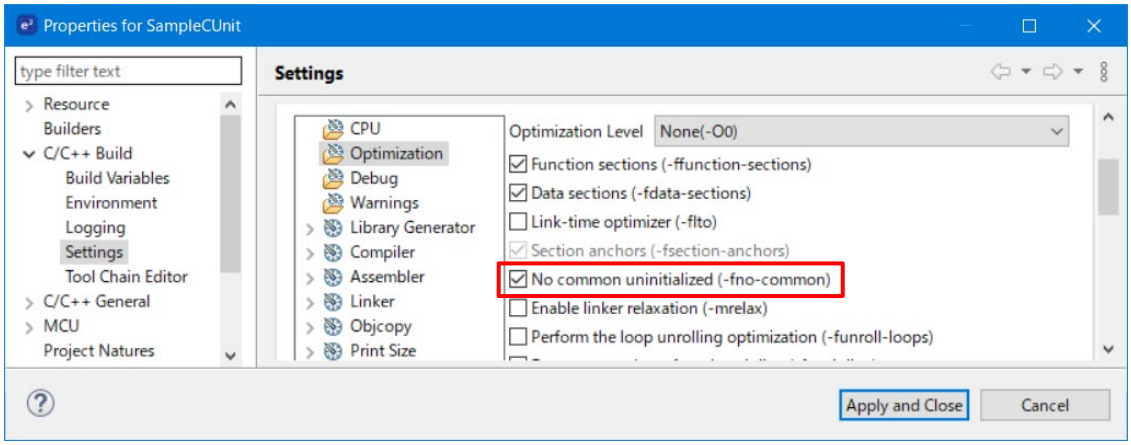

Figure 11 Check Optimization setting

13) In [Link] > [Other], [User defined options], click [Add...] button and add option "-msim". Then, click [Apply and Close] button.

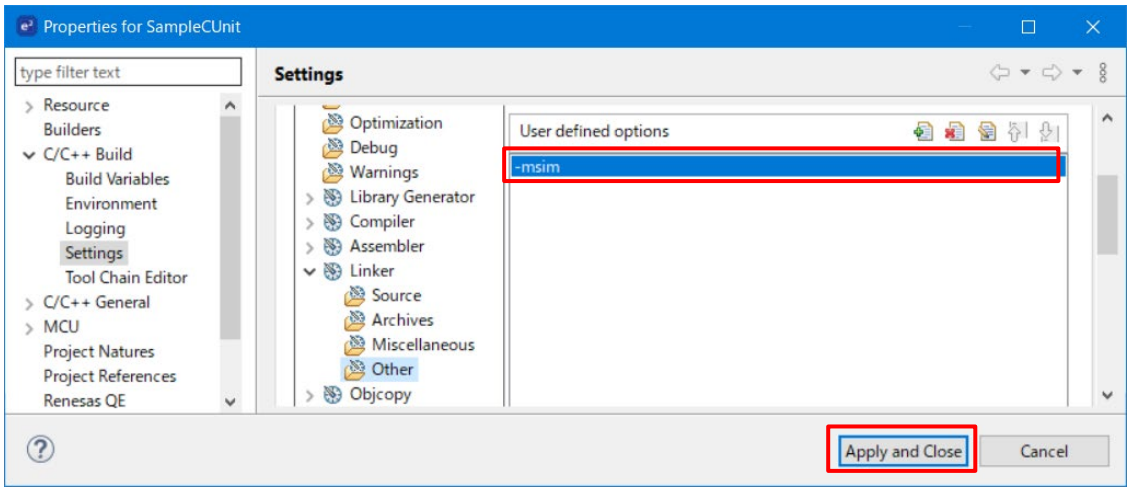

Figure 12 Add user defined option

14) Build the project.

15) To run the test harness on a GDB simulator use rx-elf-run in the Terminal view. To do this, expand the Binaries node in the Project Explorer, select the SampleCUnit.elf file, and from the context menu choose [Show In Local Terminal] > [Terminal]. The Terminal view opens in the directory containing the SampleCUnit.elf file.

16) In the Terminal view, enter "rx-elf-run SampleCUnit.elf" and press enter. The test result is displayed in the Terminal view, as shown in the figure below:

> There is "rx-elf-run.exe" in the "<GCC for Renesas RX install folder>\rx-elf\rx-elf\bin" folder. Be sure to add that folder to the "Path" environment variables.
>
> e.g.; set PATH=<GCC for Renesas RX install folder>\rx-elf\rx-elf\bin;%PATH%

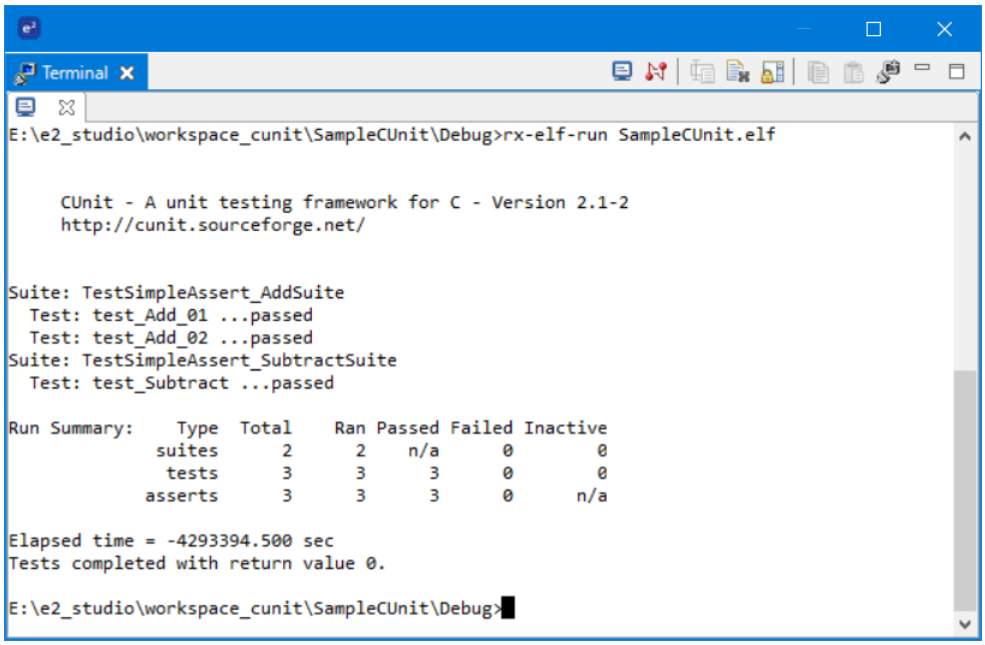

Figure 13 Executing test program in Terminal view

3. Reference information

3.1 Website and Support

- Renesas Electronics Website
  http://www.renesas.com/
- Inquiries
  http://www.renesas.com/contact/
- GNU Tools for Renesas RX/RL78
  https://llvm-gcc-renesas.com

3.2 When using other devices or compiler or debugger

This document assumes an environment that combines rx-elf-run (simulation environment for GCC RX) and printf, but in the debugger for Arm cores, console output is possible by semi-hosting function etc. In addition, even if the emulator does not have a console output function and output with printf cannot be performed, it is possible to display on the console by using "Dynamic printf".

You can see how to use "Dynamic printf" in the video on the following page.

e² studio Tips - How to Use Printf Debugging Without Changing the Source Code (Using Dynamic Printf) | Renesas

[Example]

If you create your own printf as shown below and specify "dynamic printf" there, you can get the same result as in this document.

- xprintf.h

```
#ifndef XPRINTF_H_
#define XPRINTF_H_

#define printf xPrintf
void xPrintf(const char* format, ...);

#endif
```

- xprintf.c

```
void xPrintf(const char* format, ...);

void xPrintf(const char* format, ...)
{
   static char szBuf[512];
   va_list ap;
   va_start(ap, format);

   vsprintf(szBuf, format, ap);

   va_end(ap);   /* here place Dynamic Printf as "%s",szBuf */
}
```

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct 29,2018	-	First edition issued
1.01	Jul 26, 2021	All	Update all according with e² studio 2021-07 environment.
1.02	Jul 12,2022	Page 1, 2 Page 13	- Delete the procedure for the combination with Jenkins since it is insufficient description of it. (That procedure in detail will be described by other application note.) - Add the explanation of “Dynamic printf”.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.