Integrated Development Environment e2 studio

R20AN0313EJ0100
Rev.1.00
May 16, 2014

how to use Unity in e2 studio

Introduction

Unity is a simple Unit Test Framework for Embedded C. This document describes how to use Unity in Renesas e^2 studio environment.

Contents

1. Introduction

1.1 e^2 studio with Unity

Unity is a simple Unit Test Framework for Embedded C. It consists of several files. The core Unity test framework is a single C and a couple header files, which provides functions and macros to make testing easier.

This document describes how to use Unity in Renesas e^2 studio environment. If you would like to know more about Unity, please refer to 'Unity Summary.pdf' in Unity package.

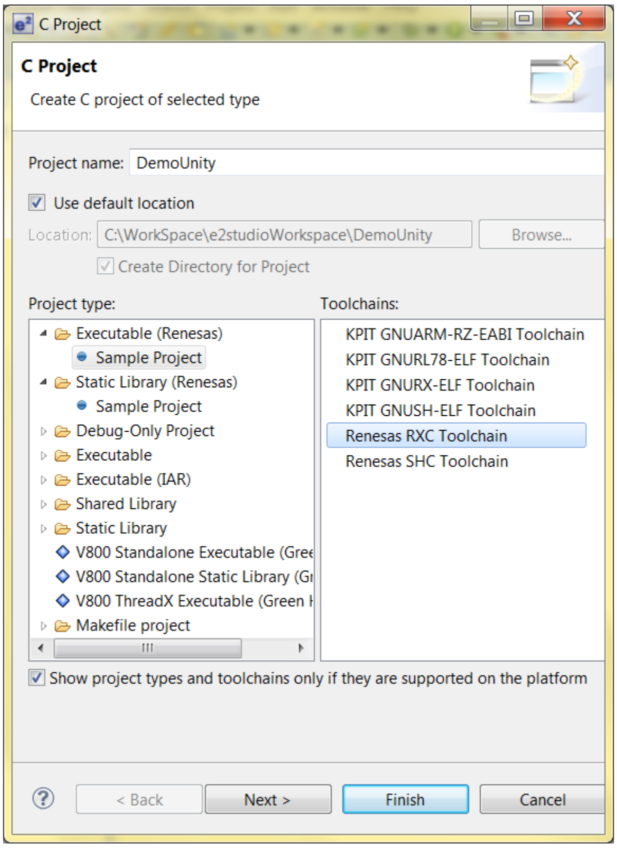
1.2 Environment

This document is described based on environment as the following:

e^2 studio:	3.0.x.x
Ruby:	1.9.3
OS:	Windows 7

2. How to use Unity with Renesas e^2 studio environment

1. Download Unity from http://sourceforge.net/projects/unity/ . Extract zip file to get Unity package.
2. Open e^2 studio, create C project with Renesas toolchain by:
   A. Select 'File' > 'New' > 'C Project'
   B. On the 'C Project' dialog, specify name of project. Select 'Executable (Renesas)' category > 'Sample Project' > 'Renesas RXC Toolchain' > Press the 'Finish' button



3. Add new source files to project. For example:

source.c (This file contains the function to be tested.)

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

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

source.h (This file contains the prototype declaration of the function to be tested.)

```
#ifndef SOURCE_H_
#define SOURCE_H_

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

#endif /* SOURCE_H_ */
```

4. Copy Unity files to project (contain unity assertion in order to use in your test).
- 'unity_internals.h', 'unity.c', 'unity.h' (See unity package folder (\unity\src) to find these files.)
5. Copy files to project (do this step to display result on Renesas Debug Virtual Console.) For these low-level interface routines, please refer to the compiler manual.
- lowlvl.src, lowsrc.c, lowsrc.h

6. Add 'testsource.c' to project (This file contains functions to test functions in 'source.c'.)

```
#include "source.h"
#include "unity.h"

// This function is called before EACH TEST.
void setUp(void)
{
  //This is run before EACH TEST.
  //If there is necessary process, describe it here.

}
// This function is called after EACH TEST.
void tearDown(void)
{
  //This is run after EACH TEST.
  //If there is necessary process, describe it here.
}

// This is a test case used to test add() function in source.c
void test_AddFunction(void)
{
  //Equal Assertion is used in this test case.
  //9 is expected value, and add(1,8) is actual return value.
  //If expected value is not same assertion occurs.
  //we can refer the Reference document for the other useful assertion
  TEST_ASSERT_EQUAL(9,add(1,8));
  TEST_ASSERT_EQUAL(10,add(2,8));
  TEST_ASSERT_EQUAL(0,add(0,0));
}

// This is a test case used to test subtract() function in source.c
void test_SubtractFunction(void)
{
  //Equal Assertion is used in this test case.
  //8 is expected value, and subtract(9,1) is actual return value.
  TEST_ASSERT_EQUAL(8,subtract(9,1));
  TEST_ASSERT_EQUAL(10,subtract(20,10));
  TEST_ASSERT_EQUAL(0,subtract(30,30));
}
```

7. This step descripts how to create a test runner (which includes ‘main’) to run the test case. As follows, there are two ways of creating the test runner: manually, or automatically.

   A. Create a test runner by manual

   Open ‘DemoUnity.c’ (call test function, and run test case)

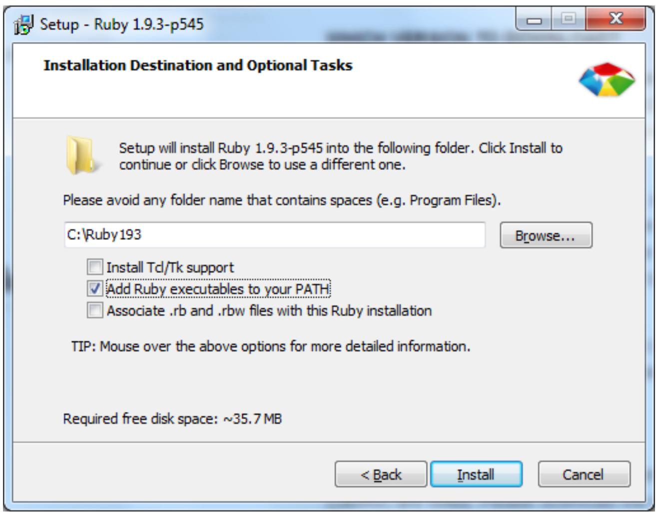
```
#include "unity.h"
char MessageBuffer[50];

extern void setUp(void);
extern void tearDown(void);

extern void test_AddFunction(void);
extern void test_SubtractFunction(void);

int main(void)
{
  Unity.TestFile = "src/testsource.c"; //get the name of test source file
  UnityBegin(); //Initialize number of tests

//Run test case test_AddFunction in testsource.c
//24 is the line number the test function starts on.
RUN_TEST(test_AddFunction, 24);

//Run test case test_SubtractFunction in testsource.c
  RUN_TEST(test_SubtractFunction, 31);

  UnityEnd(); //End Unity test. Print out summary of test results to console.
  return 0;
}
```

After this, please go to step 8.

   B. Create a test runner by automatic (using Ruby tool)

      a. Install Ruby

      - Download Ruby at: http://rubyinstaller.org/downloads/
      - Install to your system, select the ‘Add Ruby executables to your PATH’ option while installation



b. Copy Helper Script to project

- 'auto' folder (unity_2_0_113\unity\auto contains collection of Ruby scripts)

c. Create the 'prebuild.bat' file, put at 'DemoUnity/HardwareDebug/prebuild.bat'. The command in the 'preBuild.bat' will generate test runner file.

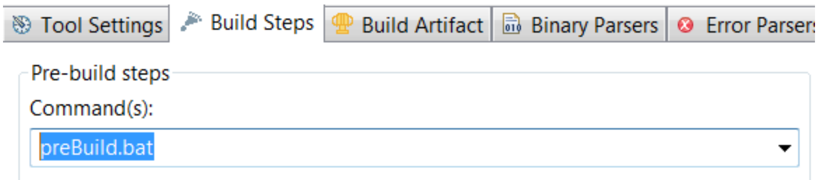
- The 'prebuild.bat' file contains the command line:

```
ruby ..\auto\generate_test_runner.rb ..\src\testsource.c ..\src\DemoUnity.c
```

('..\auto\generate_test_runner.rb': ruby script; '..\src\testsource.c': test source file contain test case; '..\src\DemoUnity.c': test runner file)

d. Add command line to e[2] studio in order to the 'preBuild.bat' file run before the project build.

Open the 'Properties of DemoUnity' panel > select the 'Settings' category > 'Build Steps' tab > Enter 'preBuild.bat' text on the 'Pre-build steps' category > select the 'OK' button to finish



8. Open ‘unity_internals.h’ file. Modify at ‘Output Method’ section
   - `Old: #define UNITY_OUTPUT_CHAR(a) putchar(a)`
   - `New: #define UNITY_OUTPUT_CHAR(a) charput(a)`

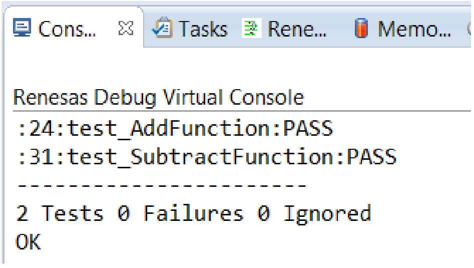
(Use ‘charput’ function to output character to the ‘Renesas Debug Virtual Console’ panel)

9. Open ‘unity.h’ file. Comment source code (for C89 only):
   - `Old: #define TEST_CASE(...)`
   - `New: //#define TEST_CASE(...)`

10. Build project

11. Connect to the debugger, and then run the program.

12. Open ‘Renesas Debug Virtual Console’ panel to see the result of debug test



3. Reference Information

3.1 Web site

http://sourceforge.net/projects/unity/

http://marketplace.eclipse.org/content/ruby-dltk/metrics

http://rubyinstaller.org/downloads/

3.2 Reference

'Unity Summary.doc' file (is Released Under a Creative Commons 3.0 Attribution Share-Alike License)

4. Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0