サードパーティーCookieが無効になっています
Hatena Blog /Test_you ✚読者になる プログ開設（無料） 管理 ログイン ヘルプ Hatena

06
24
2019

RXマイコンで、Unityによる単体テスト環境を作ってみた（後編）

RXマイコン テスト

Qiitaの記事「TDDによるマイコンのLチカ開発」を、ルネサス製のRXマイコンとIDE(e2studio)の組み合わせで、真似してみました。

1. RXマイコンで、Unityによる単体テスト環境をセットアップする。
2. シミュレータ環境、ターゲットボード環境の2通りを用意する。
3. 元の記事にあった、ホスト環境の構築、CMockの導入は省略。

- 後編では、テストフレームワーク `Unity` を導入し、テストを行います。
- 前編はこちら。RXマイコンのプロジェクト生成とprintf()での文字出力を行います。
- プロジェクト一式は、こちら(rx231_unit_test_1.zip - Google ドライブ)
  ▶ インポート方法

環境

- Board: Target Board for RX231
  - マルツで3000円くらいで購入できる
  - エミュレータ機能内蔵(E2Liteとして認識)。USB接続だけで電源供給・デバック可能
- Device: RX231(R5F52318ADFP)
- IDE: e2Stdio V7.4.0
- Compiler: CC-RX V3.01.00
- Unit Test Framework: Unity (https://github.com/ThrowTheSwitch/Unity)

テストフレームワークの導入

元の記事と同様に、Unityを導入していきます。

1.マイコンIDE上でフォルダを作成する

プロジェクト・エクスプローラーで、 `新規` > `フォルダー` を選択。以下の構造を作ります。

フォルダ構造

```
src/
    - smc_gen/              // 自動生成されたコードが格納されている
    - test/                 // テストファイルを格納 <追加>
    - unity/                // Unityのソースを格納 <追加>
    - rx231_unit_test.c     // メインのソースコード
```

2. Unityのソースコードをコピー

作成した `/src/unity/` フォルダに、Unity(https://github.com/ThrowTheSwitch/Unity)の下記ソースをドラック&ドロップで追加します。

- Unity-master/src
  `unity.c, unity.h, unity_internals.h`
- Unity-master/extras/fixture/src
  `unity_fixture.c, unity_fixture.h, unity_fixture_internals.h, unity_fixture_malloc_overrides.h`

3. テストファイルを用意する

テストファイルを `/srt/test/` 以下に用意します。それぞれ次の用途です。

- Test0.c - テストグループの定義、および、各テストケースを記述
- AllTests.c - テストで実行するテストグループを記述

/src/test/Test0.c

```
#include "../unity/unity_fixture.h"

// テストグループを定義
TEST_GROUP(Test0);

// 各テストケースの前に実行する共通処理(初期化)
TEST_SETUP(Test0)
{
}

// 各テストケースの後に実行する共通処理(後片付け)
TEST_TEAR_DOWN(Test0)
{
}

// テストケース
TEST(Test0, AlwaysFail)
{
    // 何もせず、テストを失敗させて、メッセージを出力する
    TEST_FAIL_MESSAGE("This test always fails.");
}

// テストグループで、実行するテストケースを列挙する
TEST_GROUP_RUNNER(Test0)
{
    RUN_TEST_CASE(Test0, AlwaysFail);
}
```

/src/test/AllTests.c

```
#include "../unity/unity_fixture.h"

// 実行するテストグループを列挙する
void RunAllTests(void)
{
    RUN_TEST_GROUP(Test0);
}
```

4. main関数からテストを呼ぶ

main関数からテストを呼びます。`UnityMain()`にはコマンドラインオプションが指定できます。ここでは"-v"を渡し、各テストの実行前にテスト名を出力します。

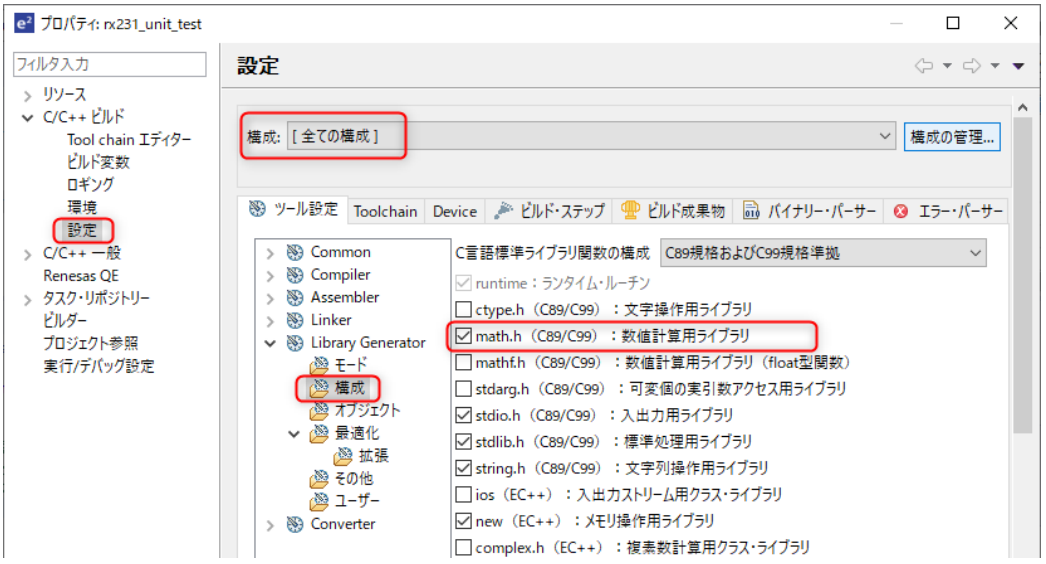
/src/rx231_unit_test.c

```
#include "r_smc_entry.h"
#include "unity/unity_fixture.h"

void main(void)
{
#if 1
    // コマンドラインオプション "-v" を指定。
    // -v  詳細(verbose)モード。各テストの実行前にテスト名を出力する。
    int argc = 2;
    const char *argv[] = {"program name", "-v"};
    extern void RunAllTests(void);
    UnityMain(argc, argv, RunAllTests);
#endif
    while(1);
}
```

5. プロジェクト設定

- プロジェクトの設定で、math.h のライブラリを生成するよう変更します[1]。
  - プロジェクトエクスプローラから、`プロパティ` > `C/C++ ビルド` > `設定` を選択
  - `ツール設定` のタブより、`Library Generator` > `構成` を選択
  - 構成を`[全ての構成]`に変更した後、`math.h` をチェック



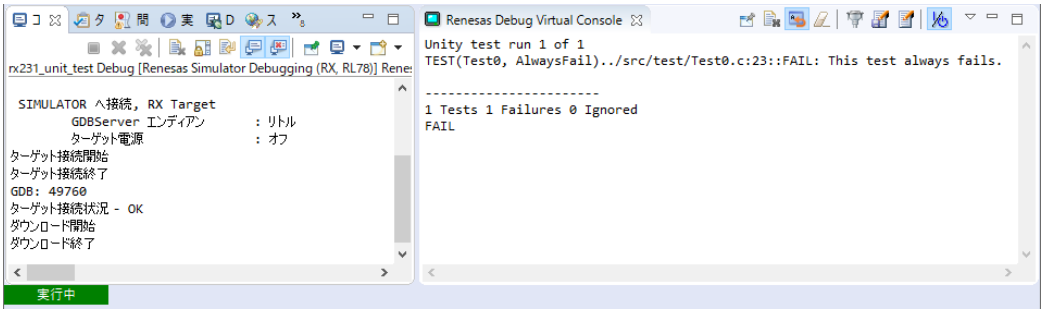
6. 実行してみる

ターゲットボード環境、シミュレータ環境のそれぞれでビルド・デバック実行し、下記が出力されればOK。

Renesas Debug Virtual Console

```
Unity test run 1 of 1
TEST(Test0, AlwaysFail)../src/test/Test0.c:23::FAIL: This test always fails.

-----------------------
1 Tests 1 Failures 0 Ignored
FAIL
```



最後に

シミュレータとRenesas Debug Virtual Consoleのおかげで単体テスト環境構築は簡単です。いつものIDE環境に＋αで気軽にテストを始められるのは良いと思います 。(これから先が大変なんでしょうけど・・・)
シミュレータ環境は通常の単体テストに。ターゲットボード環境はハードウェアの動作確認に利用できそうです。

参考にした情報

1. Qiita - @iwatake2222 さん
   - TDDによるマイコンのLチカ開発(1)
   - TDDによるマイコンのLチカ開発(2)(完)

   ⇒ 今回やってみる動機となった記事です。
2. 書籍 - テスト駆動開発による組み込みプログラミング—C言語とオブジェクト指向で学ぶアジャイルな設計
3. ルネサスのドキュメント - e2 studioでのUnityの使用方法(R20AN0313JJ0100)
   ⇒ 今回は参考にしてませんが、以前、アプリケーションノートを発行していたようです。

---

1. これを設定しないと `_FDxxxx` が見当たらないというエラーがでました。↩

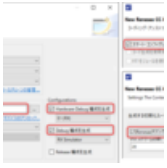
sonoka_gi 1年前

0 ツイート

関連記事

2020-07-12
Processingで、学戦都市アスタリスクのクレジット表示を真似してみた
アニメ「学戦都市アスタリスク」の２期オープニングのクレジッ...

2019-06-28
RXマイコンで、Unityを使ってハードウェアの動作確認をしてみた
前回記事「RXマイコンで、Unityを使って単体テストをやってみた...



2019-06-23
RXマイコンで、Unityによる単体テスト環境を作ってみた（前編）
Qiitaの記事「TDDによるマイコンのLチカ開発」を、ルネサス製の...

2018-11-14
ARCoreを使って、ユニティちゃんを地面で歩かせてみた
ARCoreで地面(平面)を検出し、その上でユニティちゃんを歩かせ...

コメントを書く

« RXマイコンで、UNITYによる単体テストをや…　RXマイコンで、UNITYによる単体テスト環境… »

プロフィール

sonoka_gi

電子工作やプログラミングなど、やってみたことのメモ

読者になる 1

検索

記事を検索

リンク

はてなブログ
ブログをはじめる
週刊はてなブログ
はてなブログPro

最新記事

Logicool製Webカメラ C922n のカメラ設定
C言語で、符号やサイズが異なる場合のキャスト動作を確認してみた
Processingで、学戦都市アスタリスクのクレジット表示を真似してみた
RXマイコンで、Unityを使ってハードウェアの動作確認をしてみた
RXマイコンで、Unityによる単体テストをやってみた

月別アーカイブ

▼ 2020 (3)
2020 / 12 (1)
2020 / 10 (1)
2020 / 7 (1)
▶ 2019 (4)
▶ 2018 (2)

はてなブログをはじめよう！

sonoka_giさんは、はてなブログを使っています。あなたもはてなブログをはじめてみませんか？

はてなブログをはじめる（無料）

はてなブログとは

/Test_you

Powered by Hatena Blog | ブログを報告する