

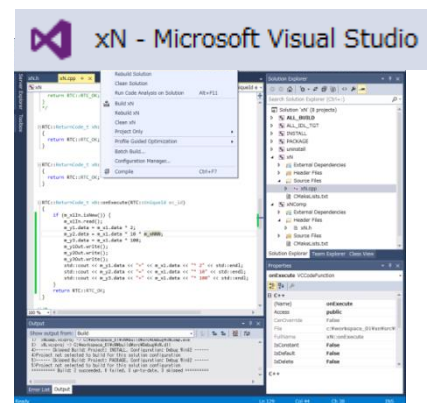
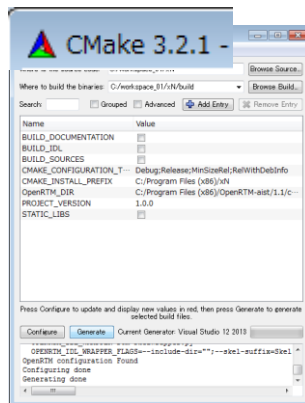
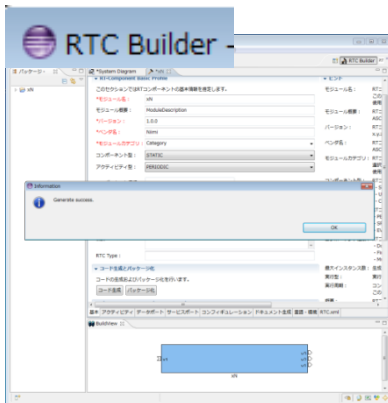
StarTno_00 説明書 02

RTC の作り方 2

(Win7, Visual Studio2013)

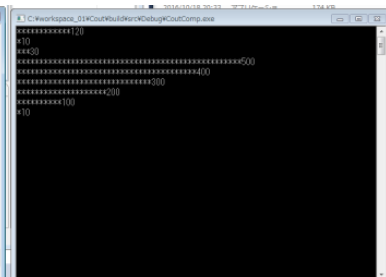
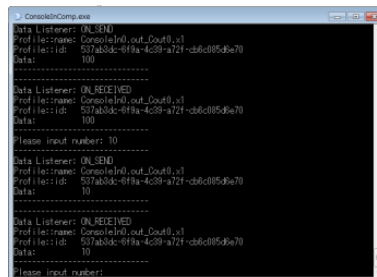
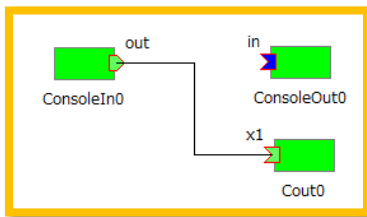
作成手順

RTC ビルダーでコンポーネントを構築し, CMake で必要なファイルを作り, VisualC++でコードを書いてビルドする. RT システムエディタで動作確認を行う.



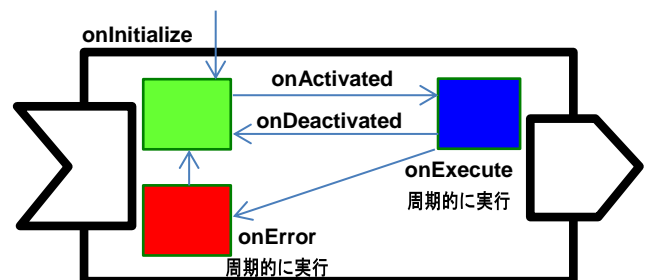
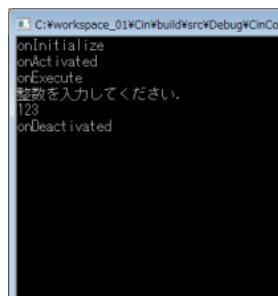
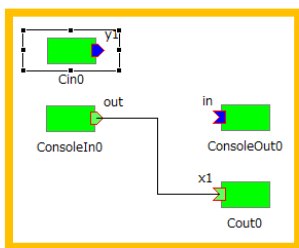
Cout (001~022)

コンソールアウトに相当する RTC. 簡易棒グラフ機能付き.



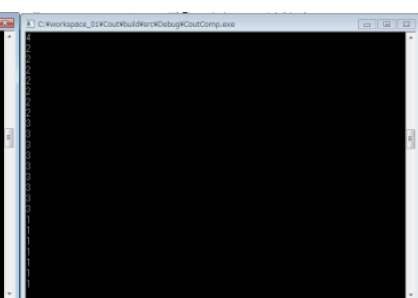
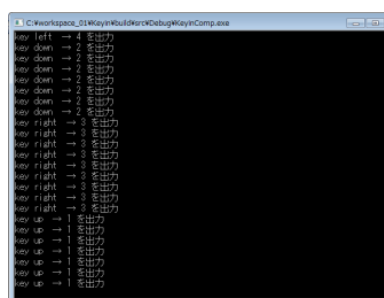
Cin (023~027)

コンソールインに相当する RTC. 状態の遷移を表示する.



Key in (028)

1, 2, a, ↑, ↓, ←, → のキーを押すと, 1, 2, 1, もと+10, もと-10, もと+1, もと-1 を出力する RTC..

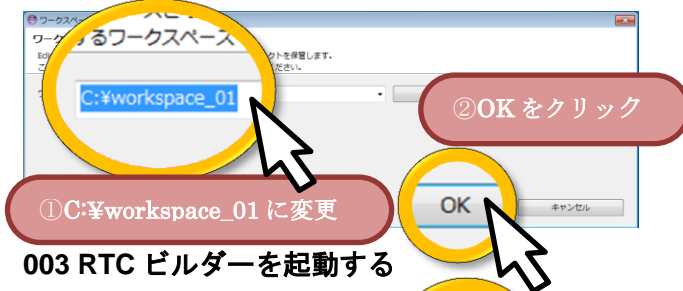


001 OpenRTP1.1.0 を起動する ①

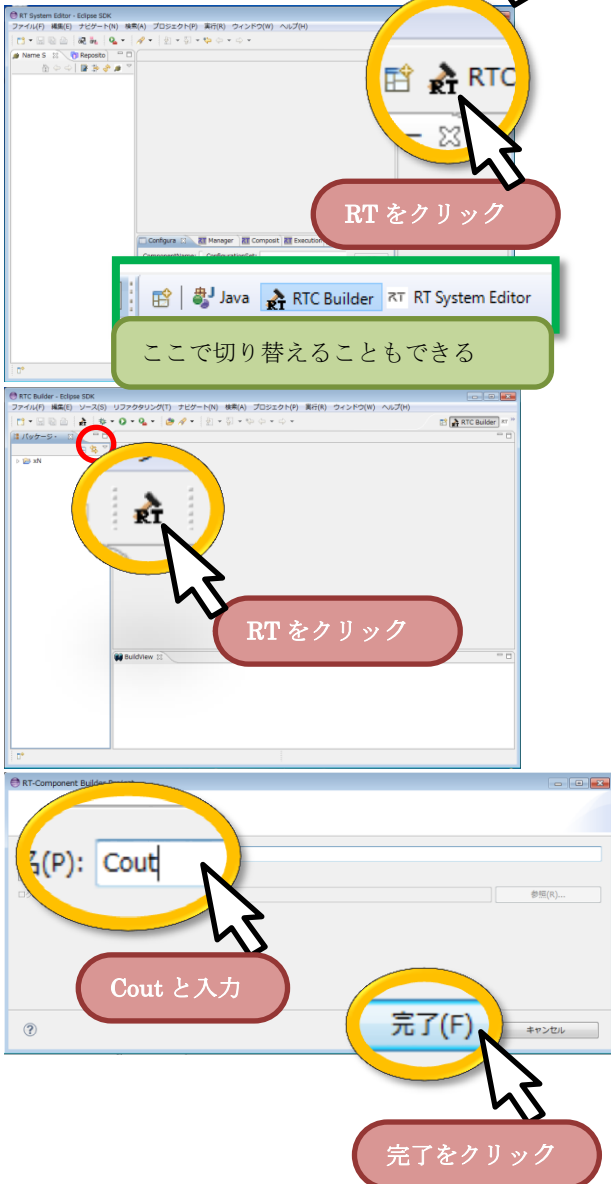


002 ワークスペースを設定する

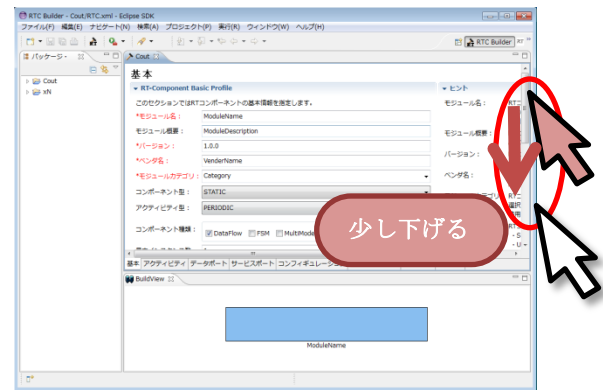
ここでは C:\workspace_01 を設定した。



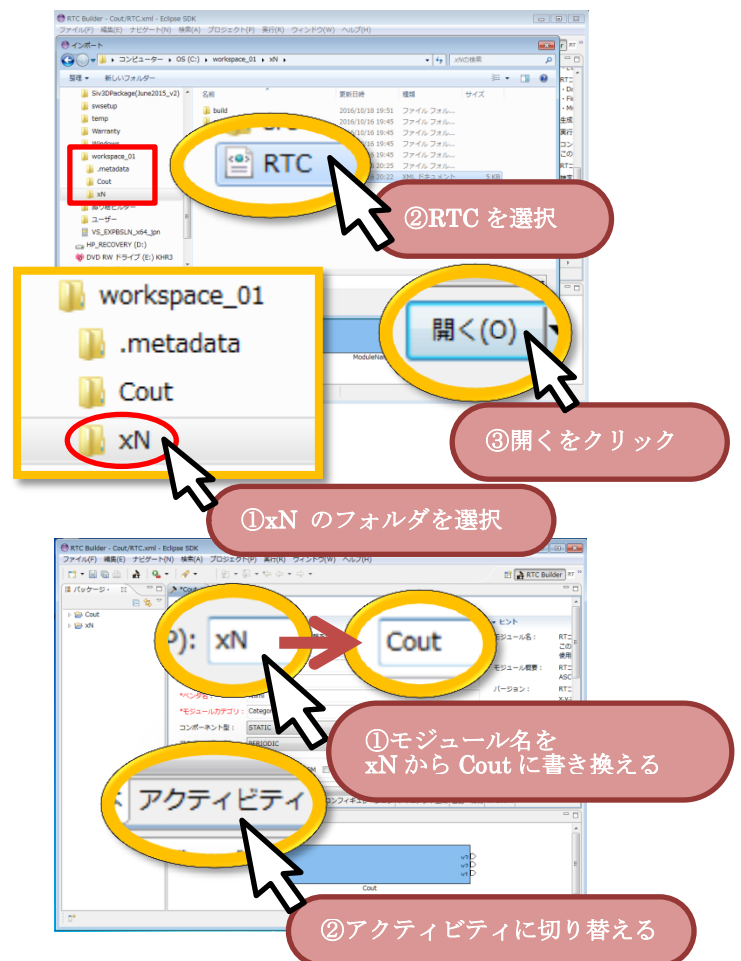
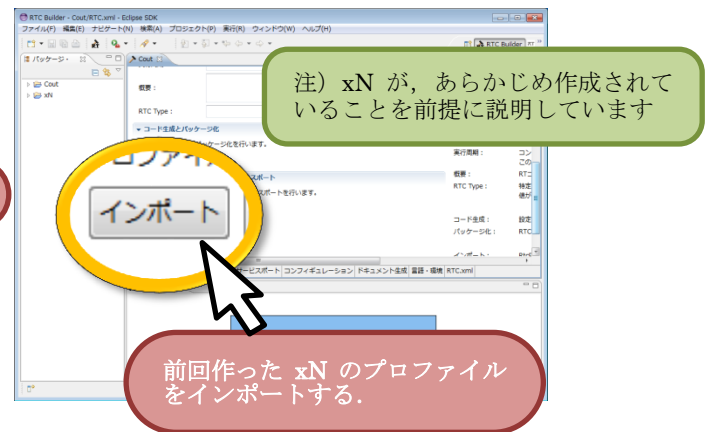
003 RTC ビルダーを起動する



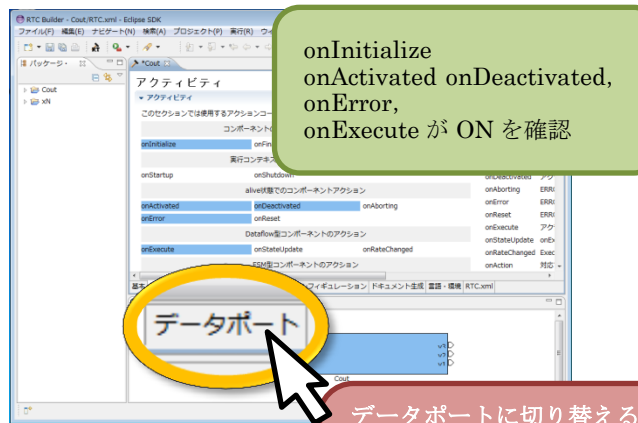
005 RTC ビルダーの画面



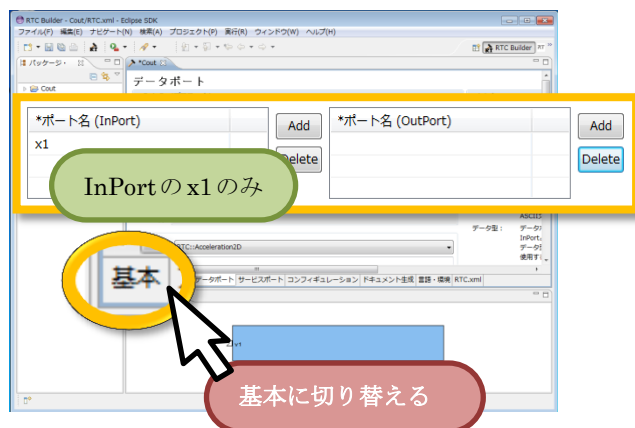
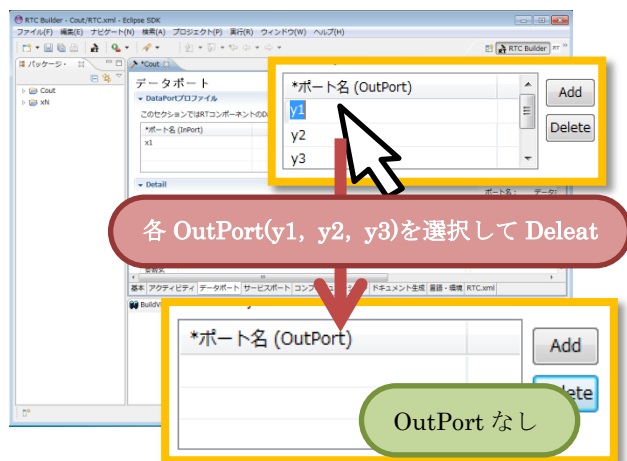
006 プロファイルをインポートする



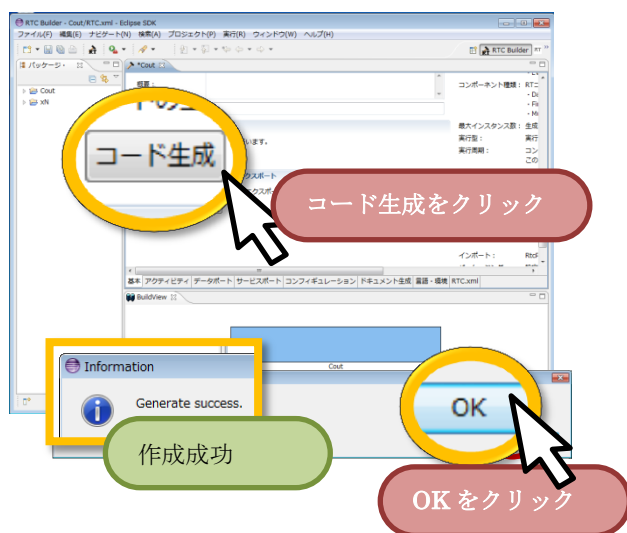
007 アクティビティを確認する. (5 か所)



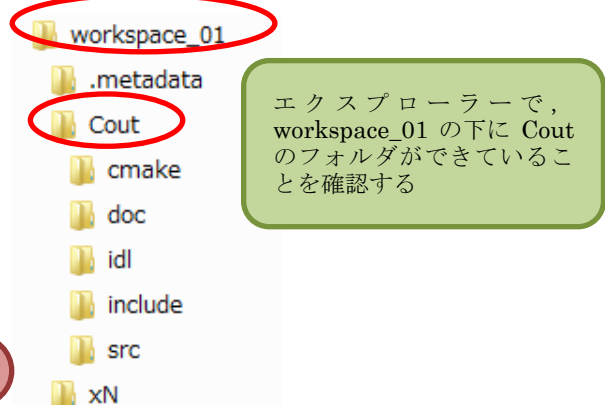
008 データポートは, x1 のみ. y1y2y3 削除



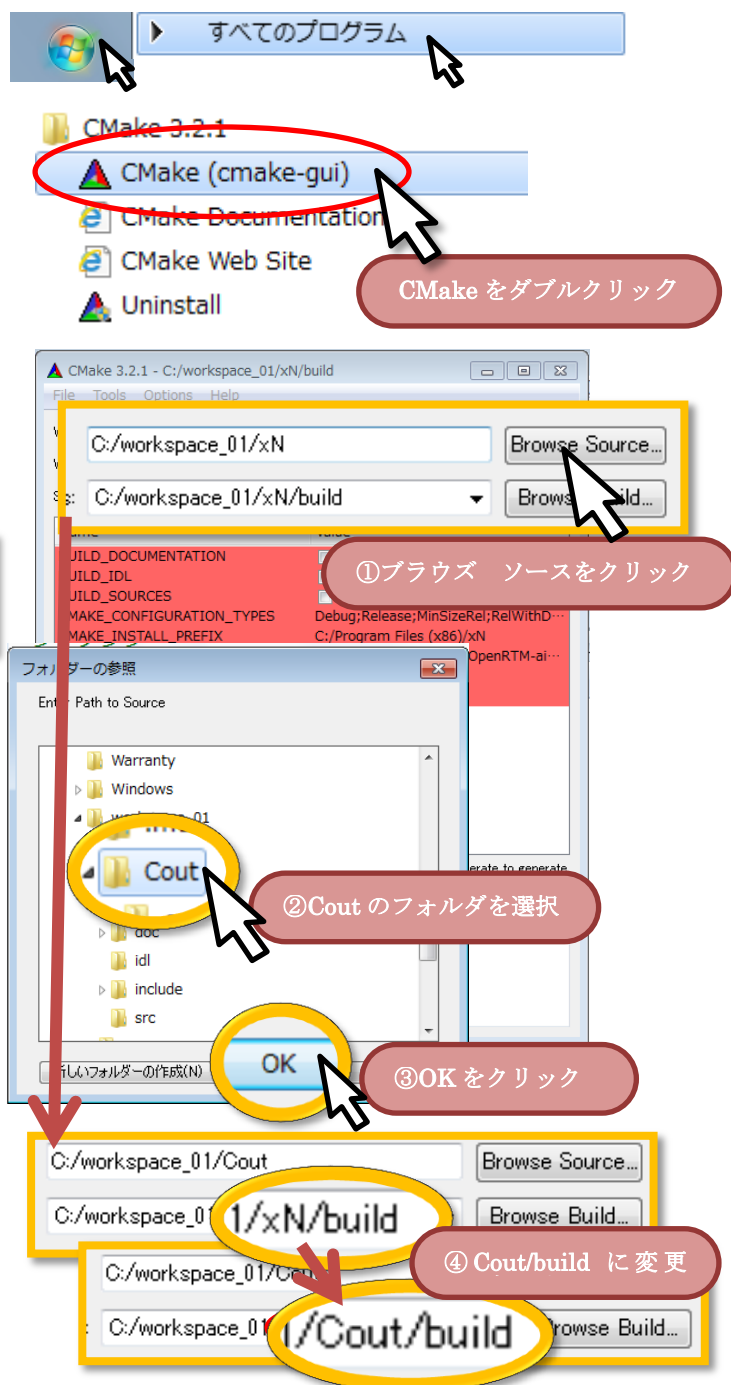
009 コード生成



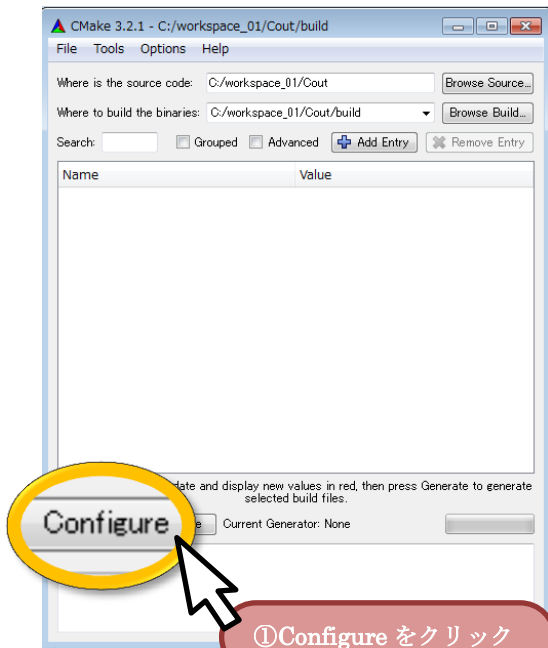
010 Cout というフォルダーが workspace_01 の下にできていることを確認する.



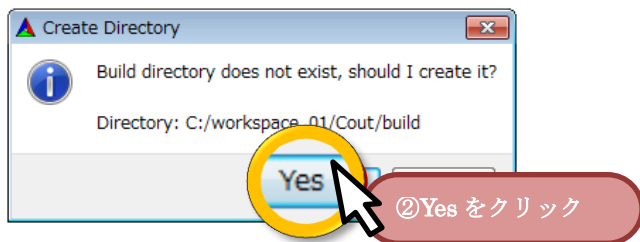
011 CMake の設定



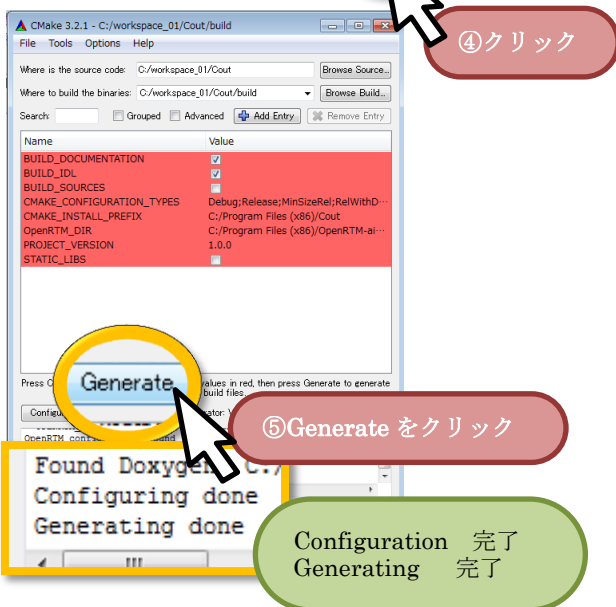
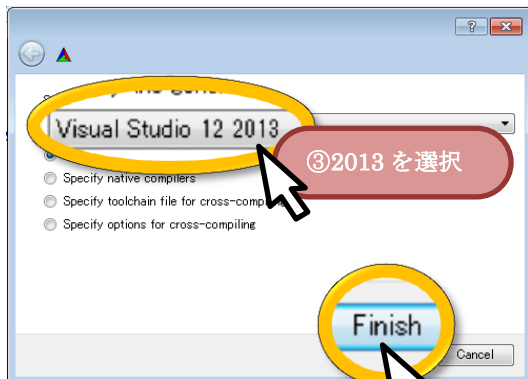
012 Cmake : Configure, Generate



build ファォルダ作成の確認。Yes.

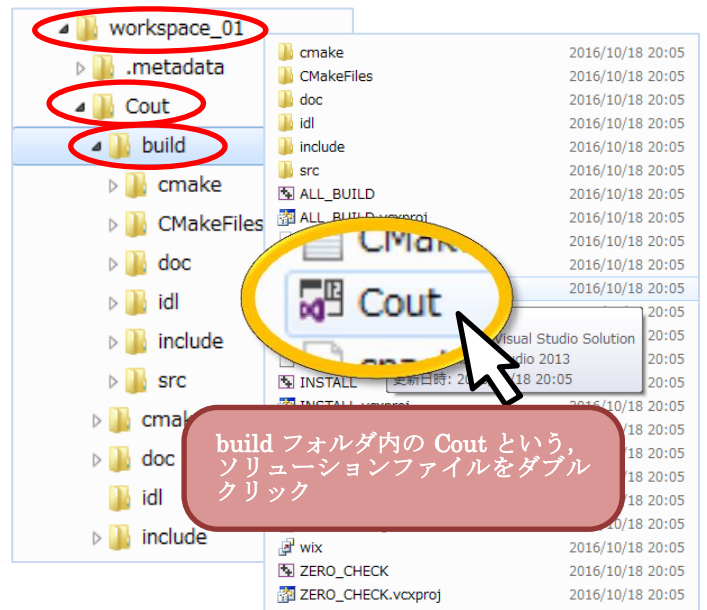


Visual Studio のバージョンの確認



013 エクスプローラーで Cout の下の build を開く.

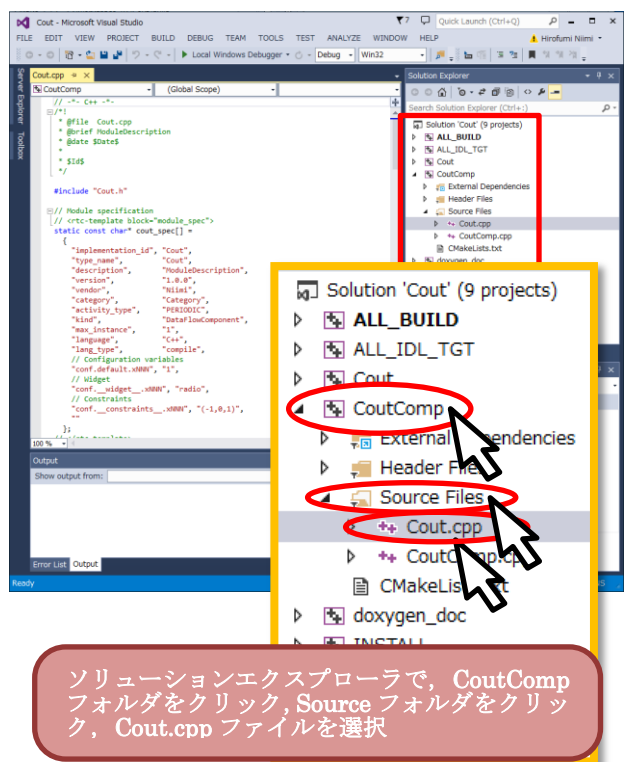
Cout のソリューションファイルをダブルクリック

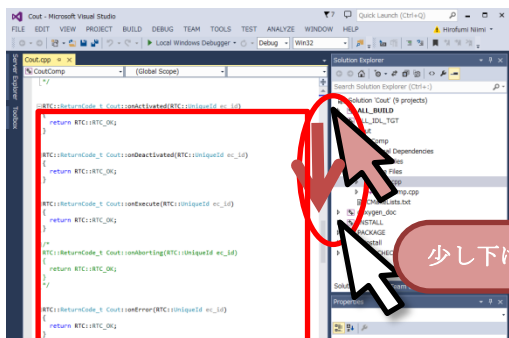


Visual Studio の起動中の画面



Cout.cpp ファイルを開く





少し下げる

```

RTC::ReturnCode_t Cout::onActivated(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}

RTC::ReturnCode_t Cout::onDeactivated(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}

RTC::ReturnCode_t Cout::onExecute(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}

/*
RTC::ReturnCode_t Cout::onPorting(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}
*/

RTC::ReturnCode_t Cout::onError(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}

```

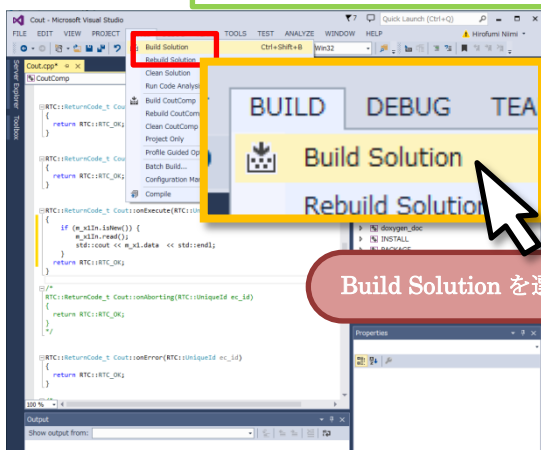
ここにコードを書く

```

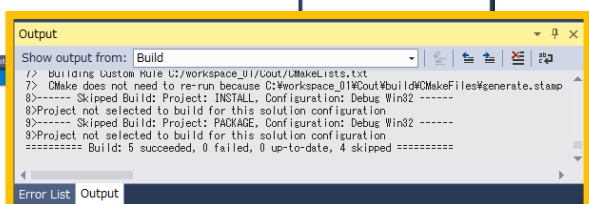
RTC::ReturnCode_t Cout::onExecute(RTC::UniqueId ec_id)
{
    if (m_x1In.isNew()) {
        m_x1In.read();
        std::cout << m_x1.data << std::endl;
    }
    return RTC::RTC_OK;
}

```

注意：エックス_いち エンド_エル

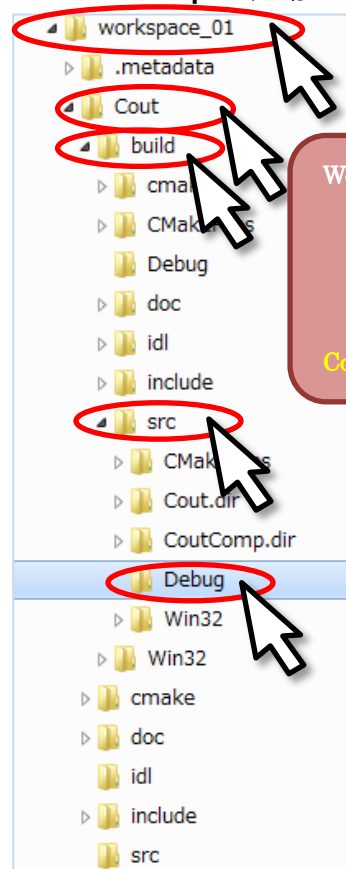


Build Solution を選択



Build succeeded(ビルド成功)を確認

014 CoutComp の起動



Workspace_01

Cout

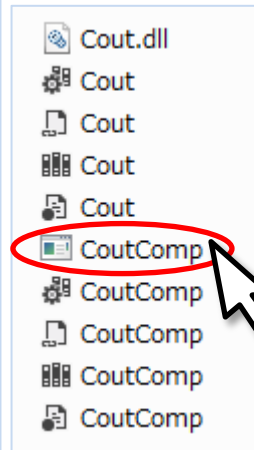
build

src

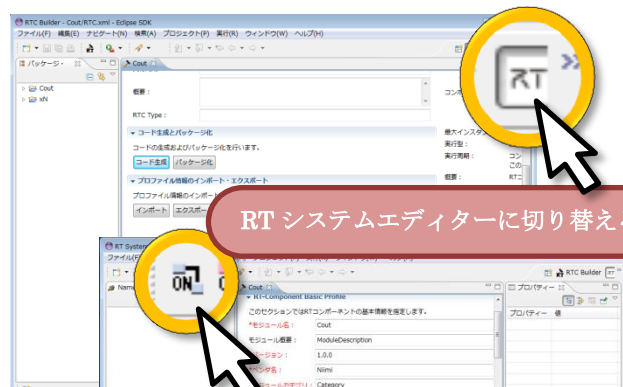
Debug

CoutComp

CoutComp をダブルクリック

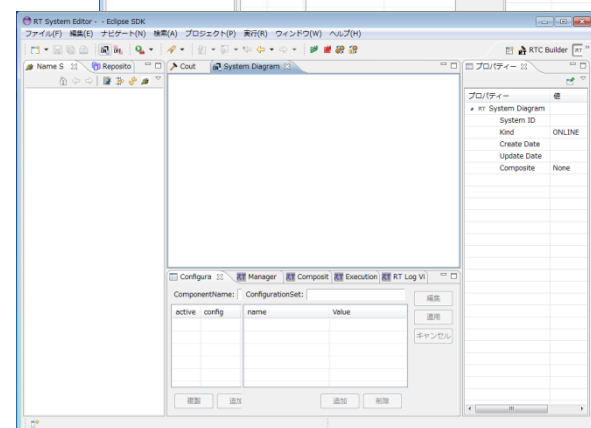


015 RT システムエディターの起動

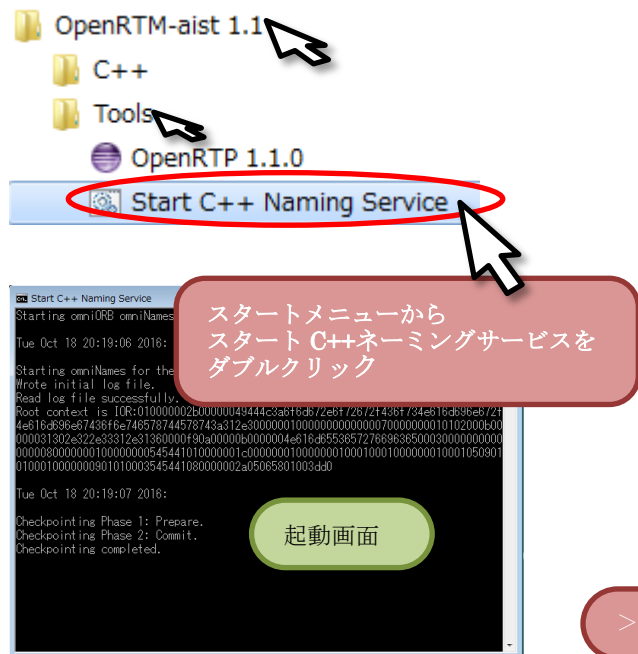


RT システムエディターに切り替える

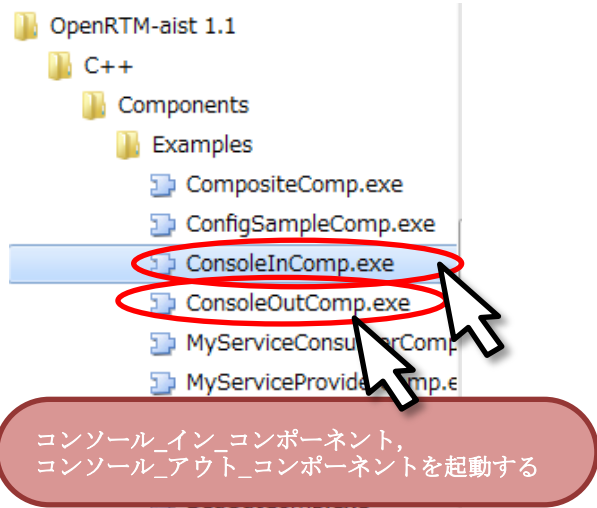
ON をクリック



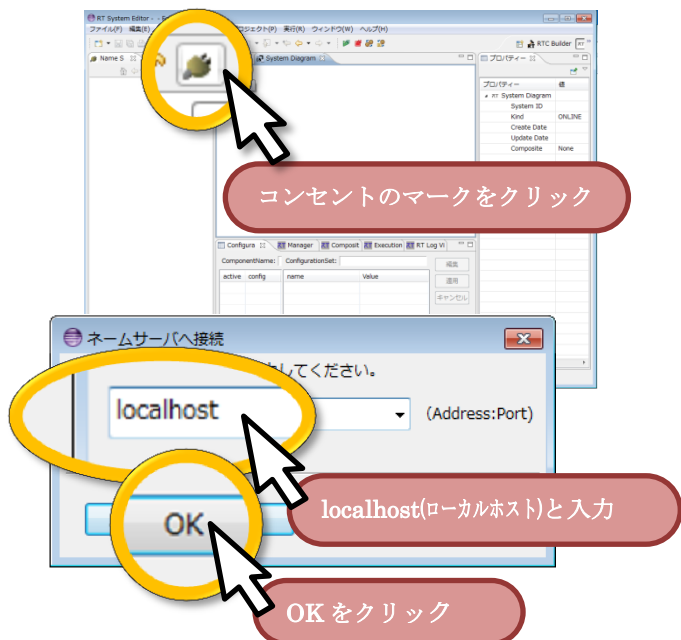
016 NamingService の起動



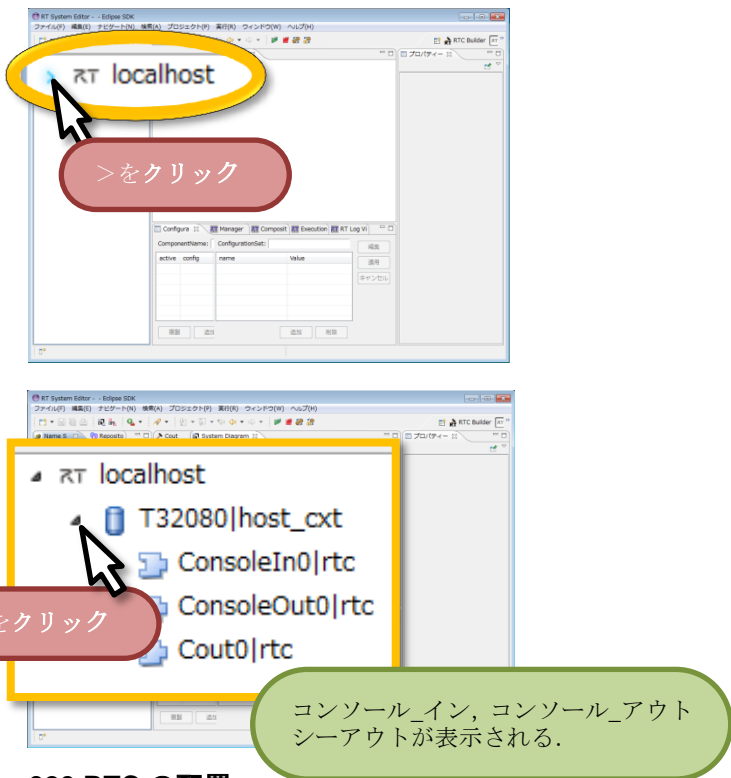
017 ConsoleInComp, ConsoleOutComp の起動



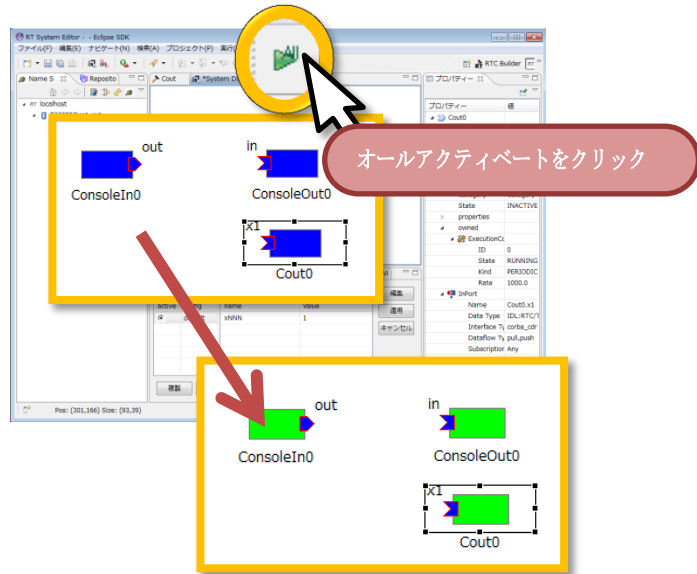
018 ネームサービスの追加



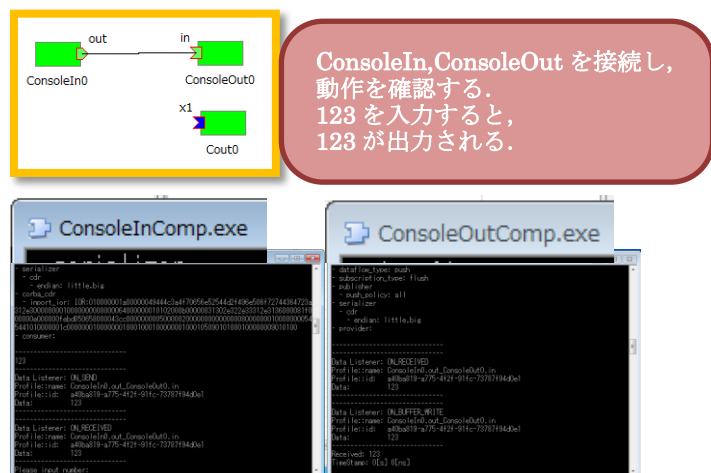
019 RTC の確認

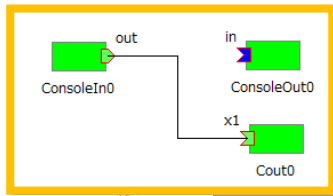


020 RTC の配置

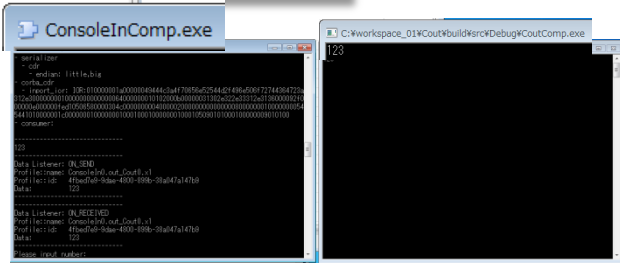


021 RTC の接続と動作確認

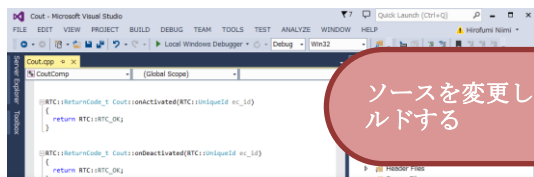




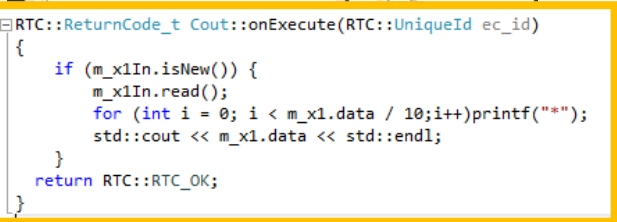
前回の接続線を削除後、ConsoleIn,Coutを接続。123 を入力すると、123 が出力される。



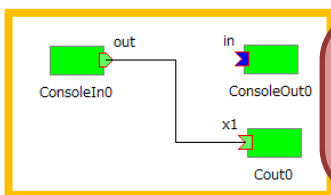
022 Cout のソースの更新



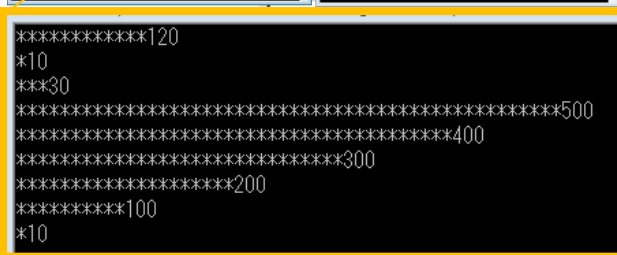
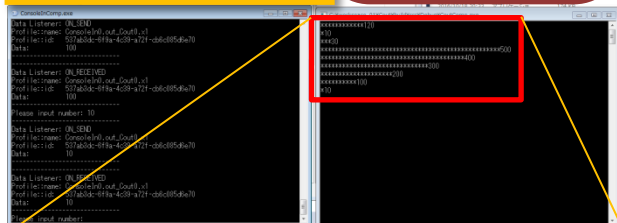
ソースを変更し、ビルドする



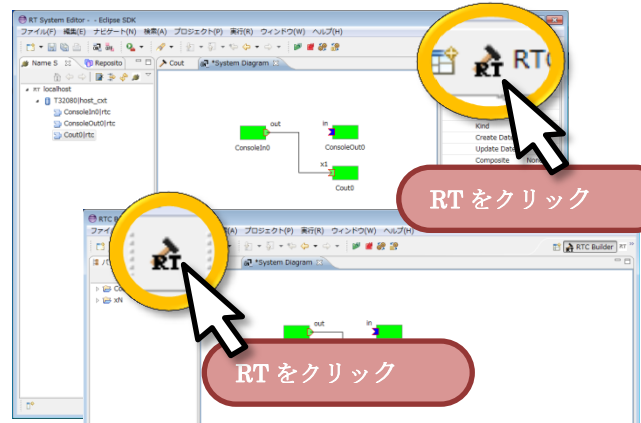
Cout を Build する。CoutComp が起動状態では、Build は失敗する。失敗した場合、CoutComp を閉じて、再び Build する。Cout を起動、接続し、動作確認を行う。10 ごとに、「*」を表示する棒グラフ。



数字を入力すると 10 で「*」が一つ表示される。棒グラフのようになる。



023 Cin コンポーネントの作成準備



RT をクリック

RT をクリック



Cin と入力

完了(F)

完了をクリック

024 RTC ビルダーで設計する

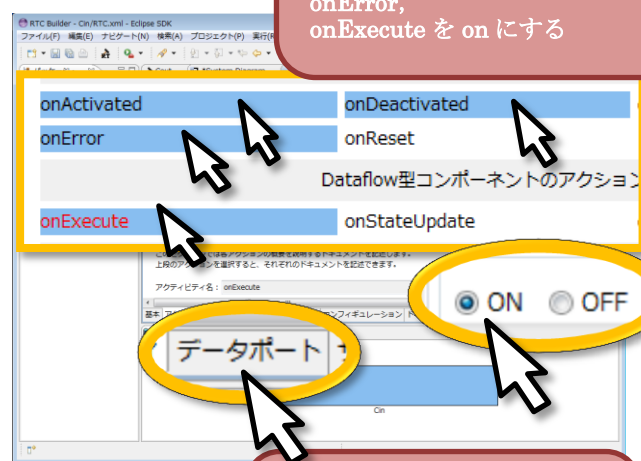


モジュール名は Cin と入力

作成者の名前

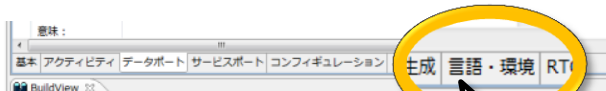
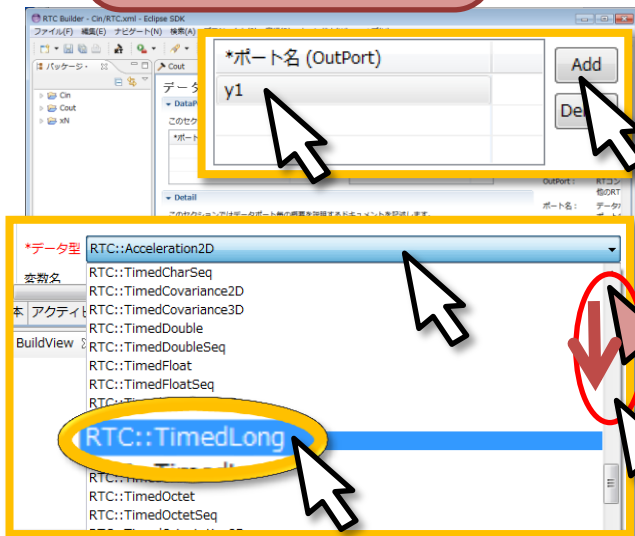
アクティビティに切り替える

onActivated onDeactivated, onError, onExecute を on にする

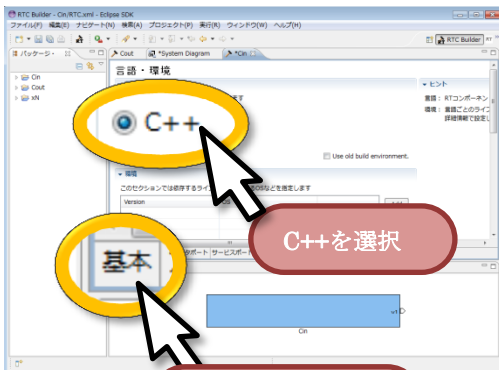


データポートに切り替える

アウトポートの Add を押し、
ポート名を y1 にする。
データ型は、TimedLong に設定



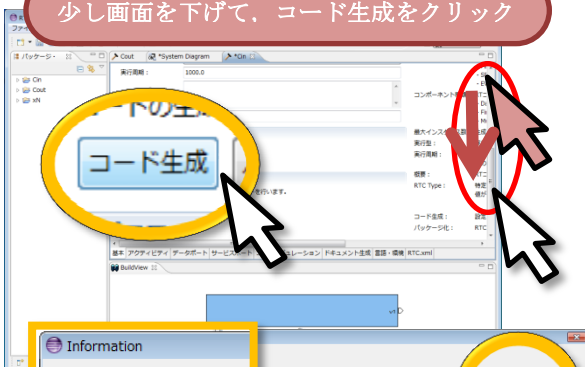
言語・環境に切り替える



C++ を選択

基本に切り替える

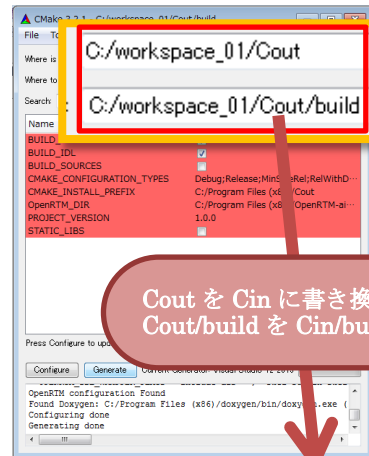
少し画面を下げて、コード生成をクリック



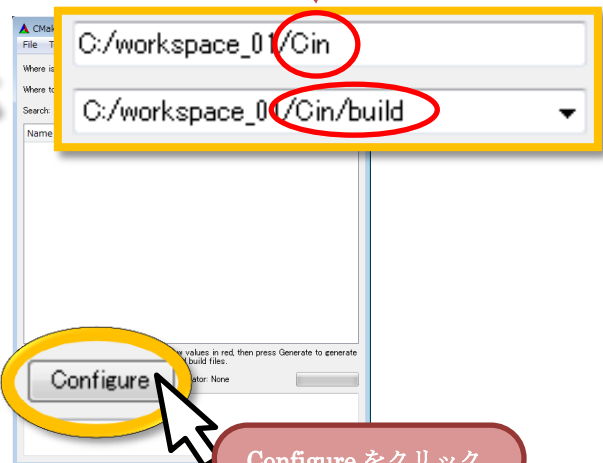
Generate success.

OK をクリック

025 CMake

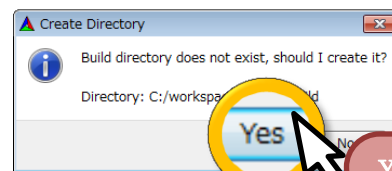


Cout を Cin に書き換える
Cout/build を Cin/build に書き換える



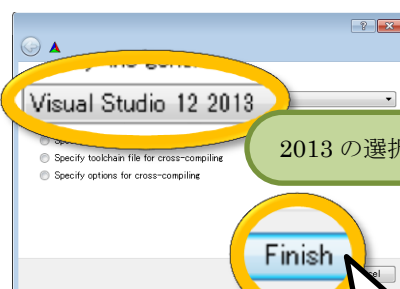
Configure をクリック

build フォルダ作成の確認。Yes.



Yes をクリック

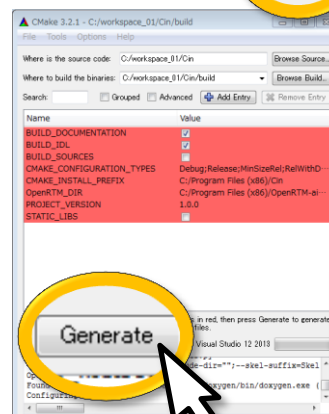
Visual Studio のバージョンの確認



2013 の選択を確認

Finish

Finish をクリック

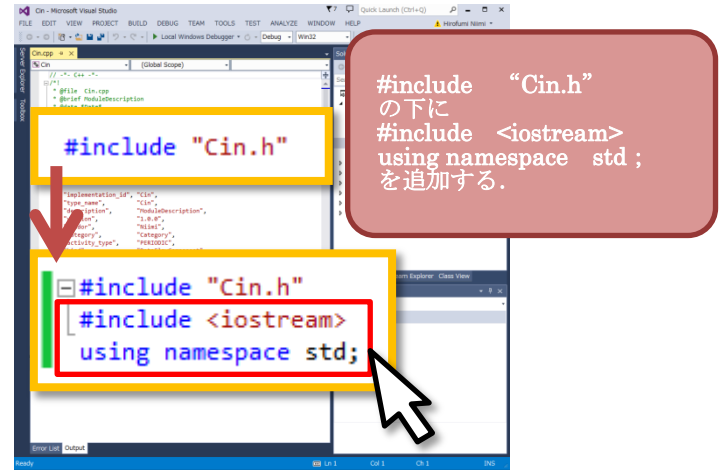
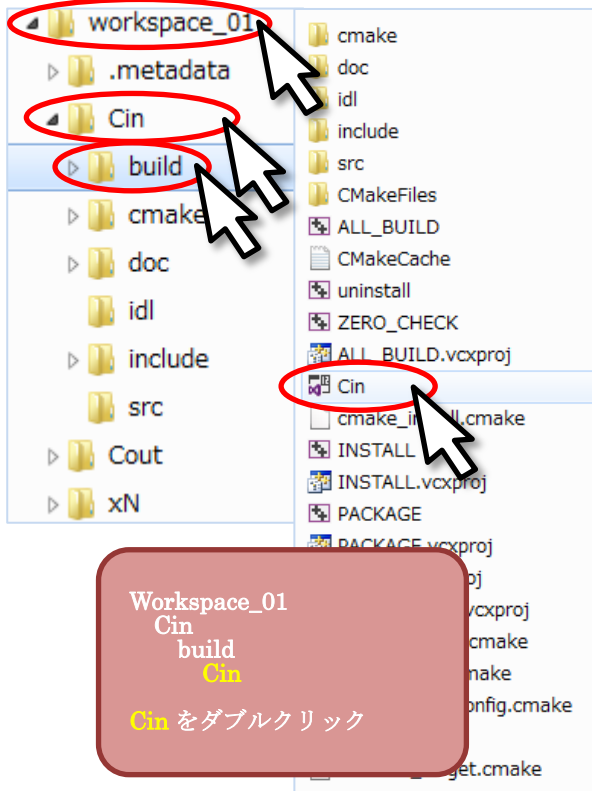


Configuration 完了
Generating 完了

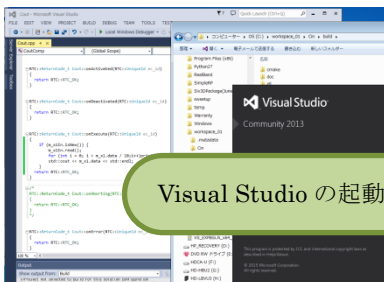
Generate をクリック

026 エクスプローラーで Cin の下の build を開く。

Cin のソリューションファイルをダブルクリック

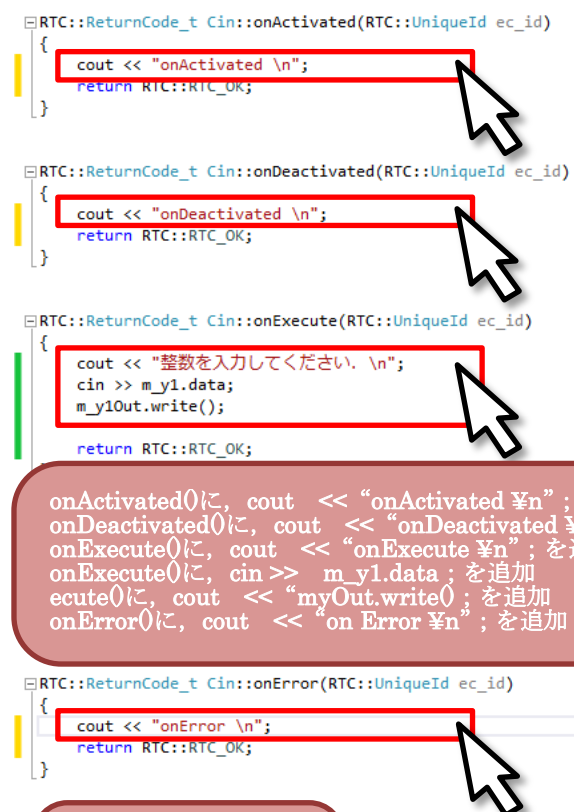
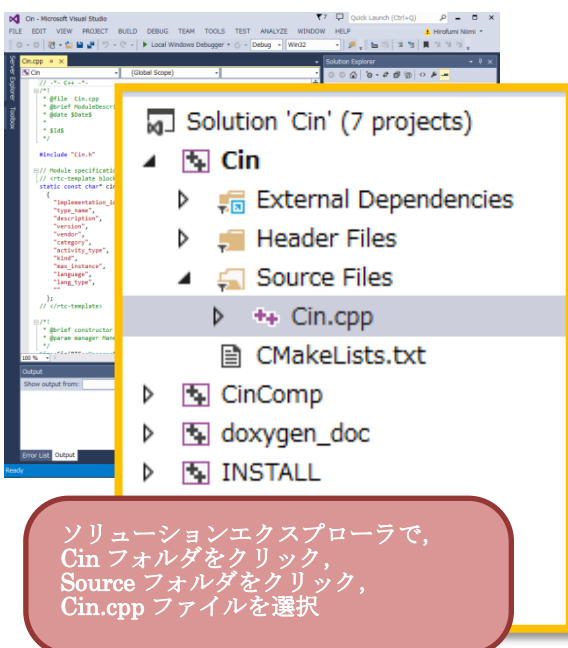


Visual Studio の起動中の画面



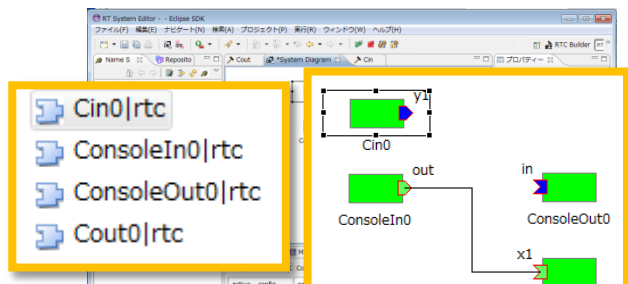
Visual Studio の起動画面

Cin.cpp ファイルを開く



Cin をビルドする

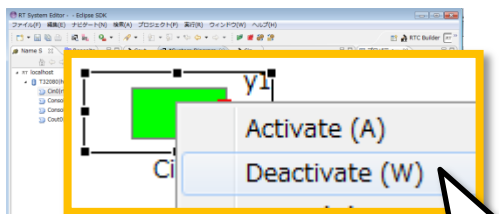
027 Cin の動作確認



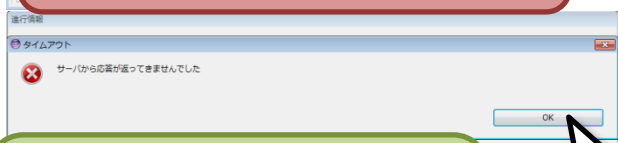
RT システムエディタに切り替え、Cin をドラッグ&ドロップ。右クリックからアクティベート。コンソールインの代わりに Cin を接続→動作確認

```
C:\workspace_01\Cin\build\src\Debug\CinComp.exe
onInitialize
onActivated
整数を入力してください。
```

状態が遷移し、onInitialize,onActivatedなどが、表示される。



右クリックから、ディアクティベートする

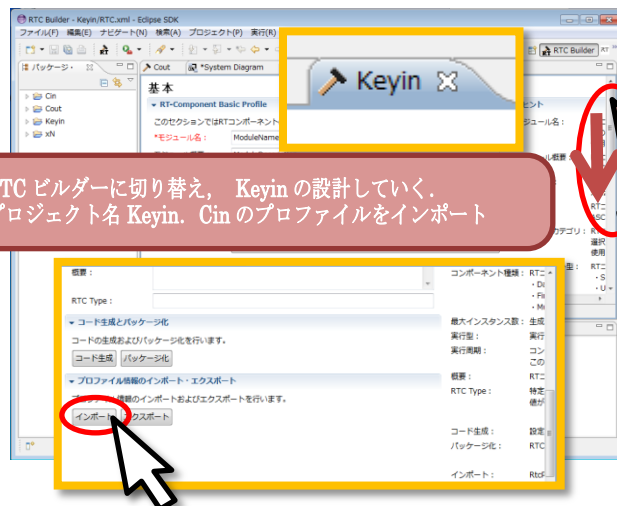


ディアクティベートが入力待ち状態のため、失敗。数字を入力すると、ディアクティベートできる。

```
C:\workspace_01\Cin\build\src\Debug\CinComp.exe
onInitialize
onActivated
onExecute
整数を入力してください。
123
onDeactivated
```

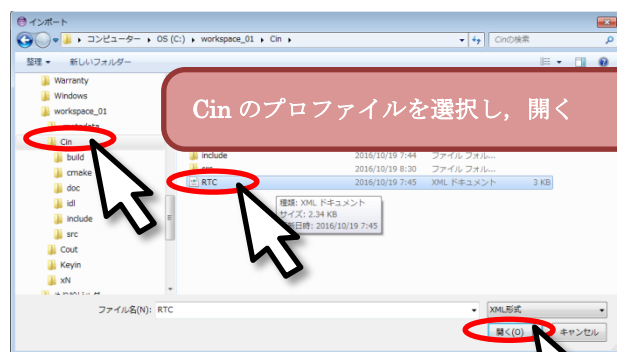
028 Keyin の設計

RTC Builder を使い、Keyin の設計していく。



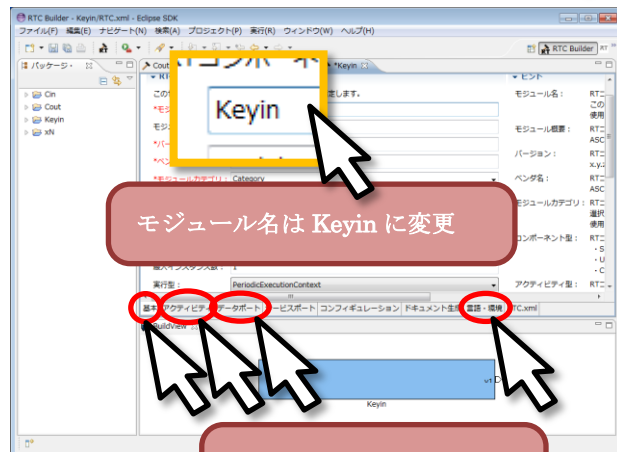
RTC ビルダーに切り替え、Keyin の設計していく。プロジェクト名 Keyin. Cin のプロファイルをインポート

Cin のプロファイルをインポート



Cin のプロファイルを選択し、開く

モジュール名を Cin→Keyin



モジュール名は Keyin に変更

各タブで設定を確認する



アクティビティを確認

データポート

DataPortプロファイル

このセクションではRTCコンポーネントのDataPort(データポート)の情報を設定します。

*ポート名 (InPort)	Add	*ポート名 (OutPort)	Add
		y1	Delete

▼ Detail

このセクションではデータポート毎の概要を上のデータポートを選択すると、それぞれのポート名: y1 (OutPort)

*データ型: RTC::TimedLong

データポートを確認
データ型を確認

言語・環境

言語

このセクションでは使用する言語を指定します

☒ C++
☐ Java
☐ Python
☐ Ruby

言語・環境を確認

実行周期: 1000.0

概要:

RTC Type:

▼ コード生成とパッケージ化

コードの生成およびパッケージ化を行います。

☒ コード生成 ☐ パッケージ化

▼ プロファイルのインポート・エクスポート

プロファイルのインポートおよびエクスポートを行います。

インポート

基本のタブに変更し、
コード生成をクリック

Information

Generate success.

OK

CMake

C:/workspace_01/Keyin

C:/workspace_01/Keyin/build

CMakeで、CinをKeyinに書き換える(2か所)
Configureをクリック。Generateをクリック

Configure

Generate

Visual Studio

Keyin.cppを表示させる

Solution 'Keyin' (9 projects)

- ALL_BUILD
- ALL_IDL_TGT
- doxygen_doc
- INSTALL
- Keyin
- External Dependencies
- Header Files
- Source Files
- Keyin.cpp
- CMakeLists.txt
- KeyinComp
- PACKAGE
- uninstall
- ZERO_CHECK

```
RTC::ReturnCode_t Keyin::onActivated(RTC::UniqueId ec_id)
{
    printf("Activated [1,2,a,↑↓++]\n");
    m_y1.data = 0;
    m_y1Out.write();
    return RTC::RTC_OK;
}
```

```
RTC::ReturnCode_t Keyin::onExecute(RTC::UniqueId ec_id)
{
    if (GetAsyncKeyState('1')){
        printf("key 1 → 1 を出力\n");
        m_y1.data = 1;
        m_y1Out.write();
    }
    if (GetAsyncKeyState('2')){
        printf("key 2 → 2 を出力\n");
        m_y1.data = 2;
        m_y1Out.write();
    }
    if (GetAsyncKeyState('A')){
        printf("key a → 1 を出力\n");
        m_y1.data = 1;
        m_y1Out.write();
    }
    if (GetAsyncKeyState(VK_UP)){
        printf("key up → 元に10足して出力\n");
        m_y1.data += 10;
        m_y1Out.write();
    }
    if (GetAsyncKeyState(VK_DOWN)){
        printf("key down → 元から10引いて出力\n");
        m_y1.data -= 10;
        m_y1Out.write();
    }
    if (GetAsyncKeyState(VK_RIGHT)){
        printf("key right → 元に1足して出力\n");
        m_y1.data += 1;
        m_y1Out.write();
    }
    if (GetAsyncKeyState(VK_LEFT)){
        printf("key left → 元から1引いて出力\n");
        m_y1.data -= 1;
        m_y1Out.write();
    }

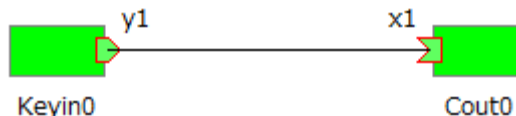
    Sleep(10);
    return RTC::RTC_OK;
}
```

入力が大変な場合、if 文の塊を一つだけ入力して動作チェックを行うと良い。
printf("この部分")は、コンソールに出力させる文字なので、簡略化しても良い。

動作確認

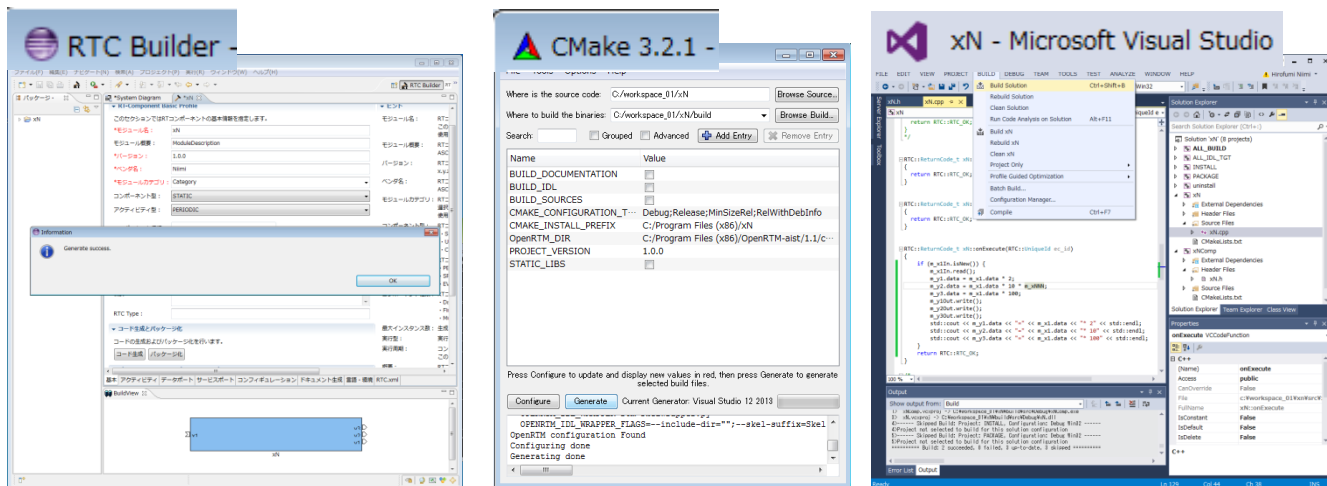
Cout0|rtc
Keyin0|rtc

RTCを接続し、アクティベート

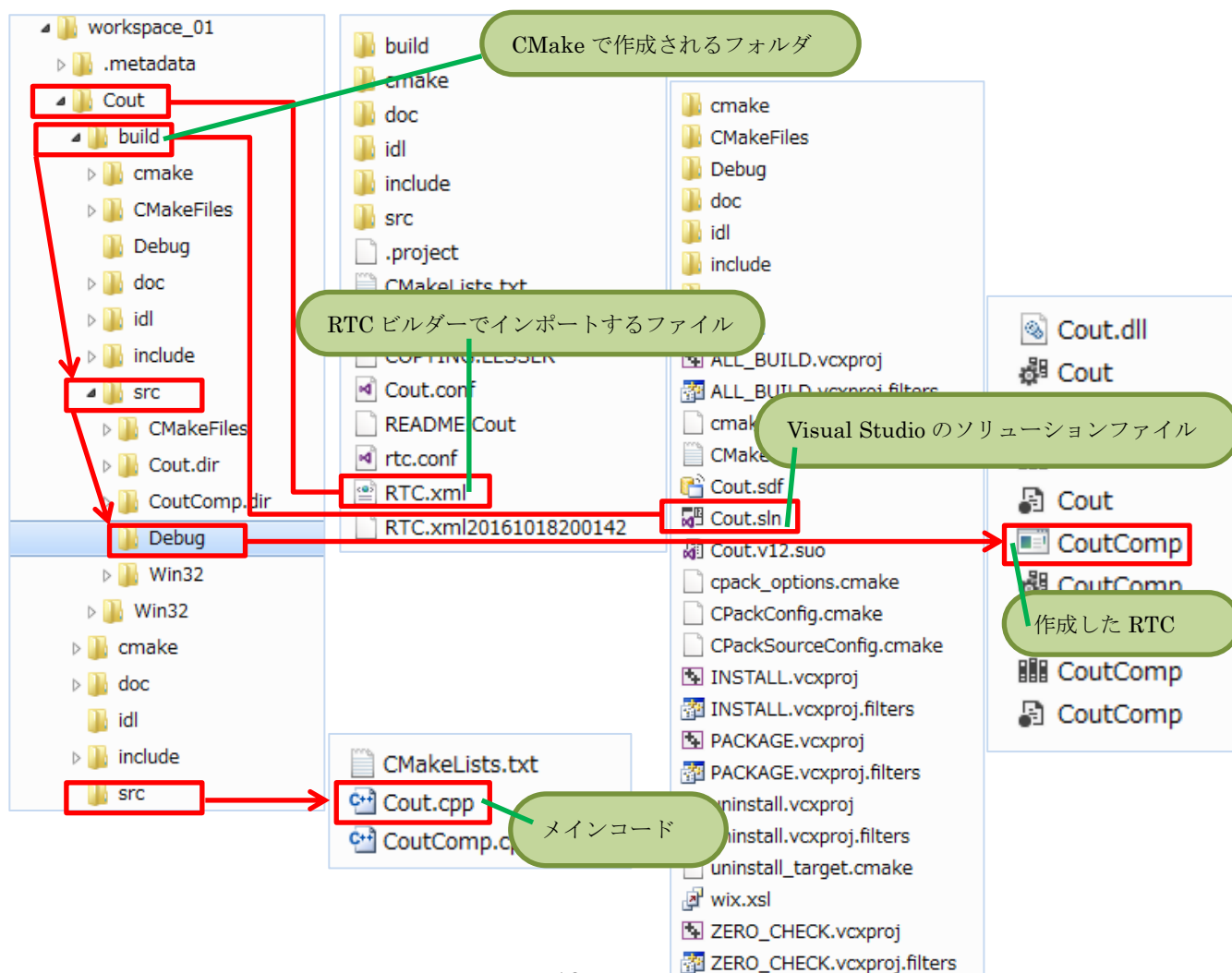


キーボードの各ボタンを押すと、それに対応した数値が出力される

RTC ビルダーでコンポーネントを構築し、CMake で必要なファイルを作り、VisualC++でコードを書いてビルドする。RT システムエディタで操作確認を行う。この操作を 3 回練習することにより、RTC 作成の基本手順が身につくように工夫した。RTC ビルダーでプロファイルをインポートする練習は、以前に作成した RTC に新たにポートを追加する場合に、使うことにできる方法である。



ソリューションファイルや作成した RTC の場所を下記に示す。



各コードの説明

Cout : コンソールアウトに相当する RTC. 簡易棒グラフ機能付き.

x1In に新しい入力があったら, x1In を読む. "*"を出力する for 文. 10 で "*" を 1 つ表示するとともに, x1 の値を出力する.

```
RTC::ReturnCode_t Cout::onExecute(RTC::UniqueId ec_id)
{
    if (m_x1In.isNew()) {
        m_x1In.read();
        for (int i = 0; i < m_x1.data / 10; i++) printf("*");
        std::cout << m_x1.data << std::endl;
    }
    return RTC::RTC_OK;
}
```

Cin(021~027)

コンソールインに相当する RTC. 状態の遷移を表示する.

#include<iostream>で標準入出力関数が使えるようにする.

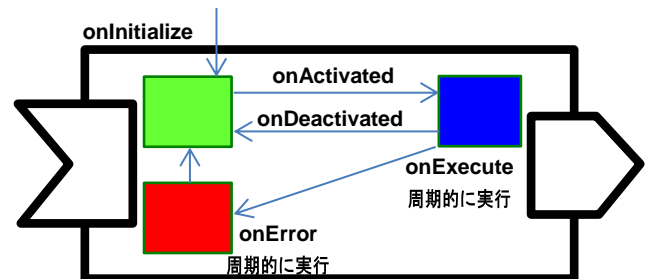
using namespace std 使う名前空間の指定.

cout << "onInitialize ¥n";は onInitialize が実行されたとき, コンソールに表示する. onActivated, onDeactivated, onError も同様に状態を表示できるようにする.

cout << "整数を入力してください. ¥n"; 整数を入力してくださいと表示.

cin >> m_y1.data; データの入力待ち状態. 入力値は y1 に渡される.

m_y1Out.write(); y1 の値を出力する.



```
// -*- C++ -*-
/*!
 * @file Cin.cpp
 * @brief ModuleDescription
 * @date $Date$
 * $Id$
 */
```

```
#include "Cin.h"
#include <iostream>
using namespace std;
```

```
RTC::ReturnCode_t Cin::onInitialize()
{
    // Registration: InPort/OutPort/Service
    // <rtc-template block="registration">
    // Set InPort buffers

    // Set OutPort buffer
    cout << "onInitialize ¥n";
    addOutPort("y1", m_y1Out);

    // Set service provider to Ports

    // Set service consumers to Ports

    // Set CORBA Service Ports

    // </rtc-template>

    // <rtc-template block="bind_config">
    // </rtc-template>

    return RTC::RTC_OK;
}
```

```
RTC::ReturnCode_t Cin::onActivated(RTC::UniqueId ec_id)
{
    cout << "onActivated ¥n";
    return RTC::RTC_OK;
}

RTC::ReturnCode_t Cin::onDeactivated(RTC::UniqueId ec_id)
{
    cout << "onDeactivated ¥n";
    return RTC::RTC_OK;
}

RTC::ReturnCode_t Cin::onExecute(RTC::UniqueId ec_id)
{
    cout << "整数を入力してください. ¥n";
    cin >> m_y1.data;
    m_y1Out.write();

    return RTC::RTC_OK;
}

/*
RTC::ReturnCode_t Cin::onAborting(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}
*/

RTC::ReturnCode_t Cin::onError(RTC::UniqueId ec_id)
{
    cout << "onError ¥n";
    return RTC::RTC_OK;
}
```


Keyin (028)

1, 2, a, ↑, ↓, ←, → のキーが押されると, 1, 2, 1, もと + 10, もと - 10, もと + 1, もと - 1 を出力する RTC. キーが押されると, TimedLong 型の数値を出力する. 1 が押されると 1 が出力し, 2 が押されると 2 を出力する. a が押されると 1 を出力する. 同じ要領で, 3, 4, 5 や b, c, d を押すと数値を出力するプログラムをつくることができる. ↑を押すと元に値に 10 足された値, ↓を押すと元の値から 10 引かれた値が出力される. →を押すと元に値に 1 足された値, ←を押すと元の値から 1 引かれた値が出力される.

```
RTC::ReturnCode_t Keyin::onActivated(RTC::UniqueId ec_id)
{
    printf("Activated [1,2,a,↑↓←→]\n");
    m_y1.data = 0;
    m_y10Out.write();
    return RTC::RTC_OK;
}
```

```
RTC::ReturnCode_t Keyin::onExecute(RTC::UniqueId ec_id)
{
    if (GetAsyncKeyState('1')){
        printf("key 1 → 1 を出力\n");
        m_y1.data = 1;
        m_y10Out.write();
    }
    if (GetAsyncKeyState('2')){
        printf("key 2 → 2 を出力\n");
        m_y1.data = 2;
        m_y10Out.write();
    }
    if (GetAsyncKeyState('A')){
        printf("key a → 1 を出力\n");
        m_y1.data = 1;
        m_y10Out.write();
    }
    if (GetAsyncKeyState(VK_UP)){
        printf("key up → 元に10足して出力\n");
        m_y1.data += 10;
        m_y10Out.write();
    }
    if (GetAsyncKeyState(VK_DOWN)){
        printf("key down → 元から10引いて出力\n");
        m_y1.data -= 10;
        m_y10Out.write();
    }
    if (GetAsyncKeyState(VK_RIGHT)){
        printf("key right → 元に1足して出力\n");
        m_y1.data += 1;
        m_y10Out.write();
    }
    if (GetAsyncKeyState(VK_LEFT)){
        printf("key left → 元から1引いて出力\n");
        m_y1.data -= 1;
        m_y10Out.write();
    }

    Sleep(10);
    return RTC::RTC_OK;
}
```

RTC で使用するデータの型

今回はすべて, TimedLong 型を使用した.