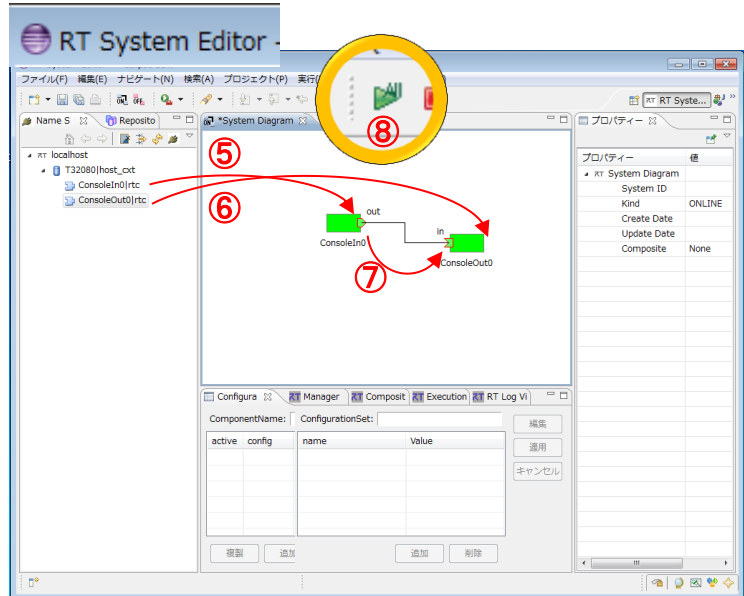
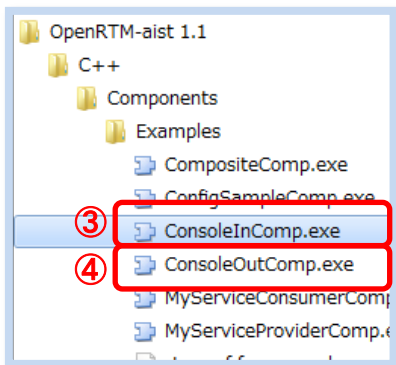
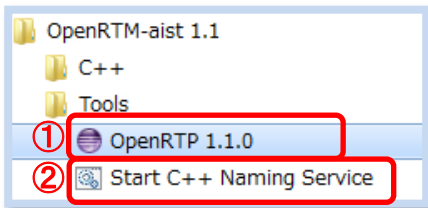


RTC の使い方・作り方 1

(Win7, Visual Studio2013)

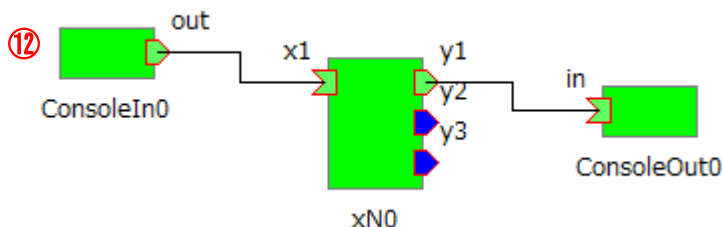
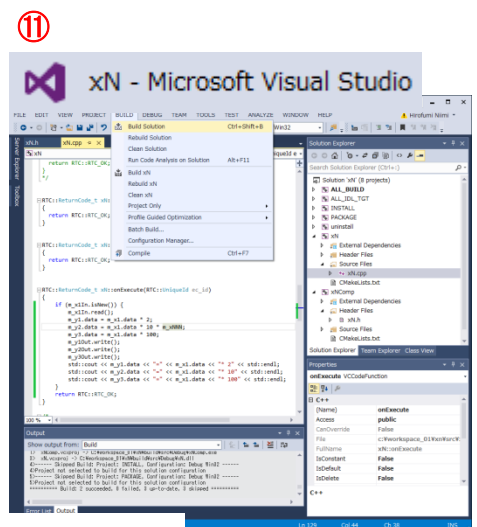
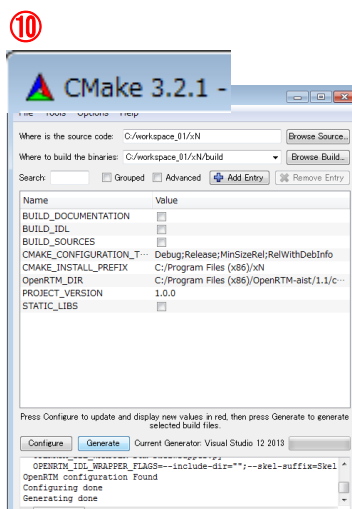
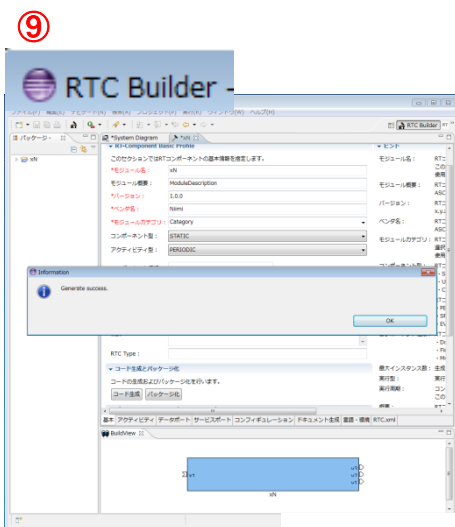
RTC の使い方 (001~031)

コンソールイン, コンソールアウトを使い, 動作確認する. ①②③④を起動し, RT システムエディター上に, RTC を配置⑤⑥・接続⑦・All アクティブート⑧する.



RTC の作り方 (032~067)

入力された値の 2 倍, 10 倍, 100 倍を出力する RTC を作成する. RTC ビルダーで設計し⑨, CMake でファイルを作り⑩, VisualC++でコードを書いてビルド⑪. 動作確認⑫.



RTC の使い方 (001~031)

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。



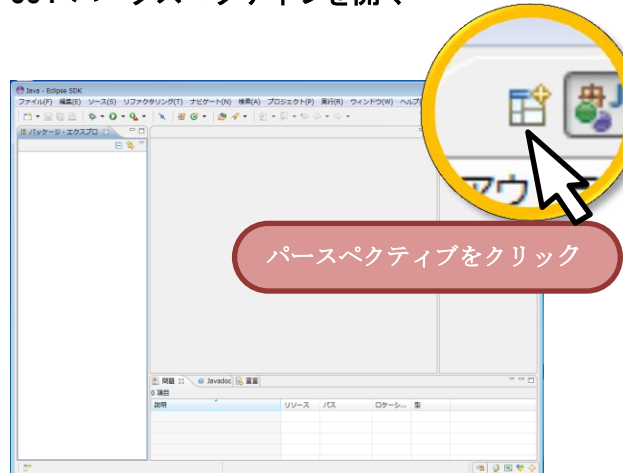
003 初期画面

インストール直後のみ

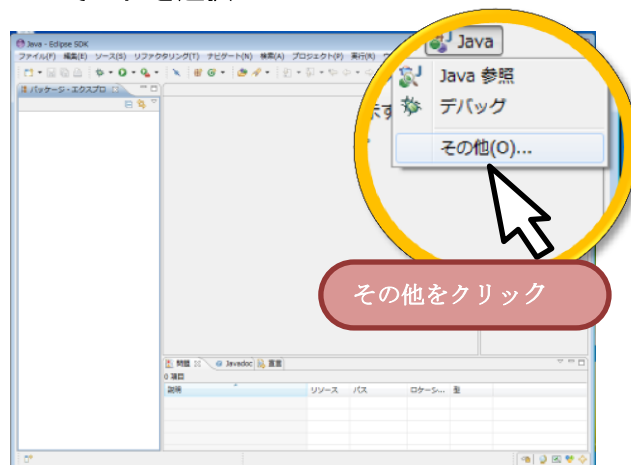
ワークベンチをクリック



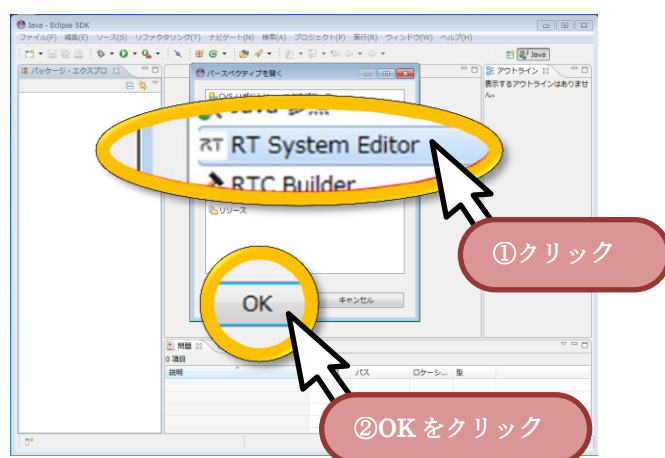
004 パークスペクティブを開く



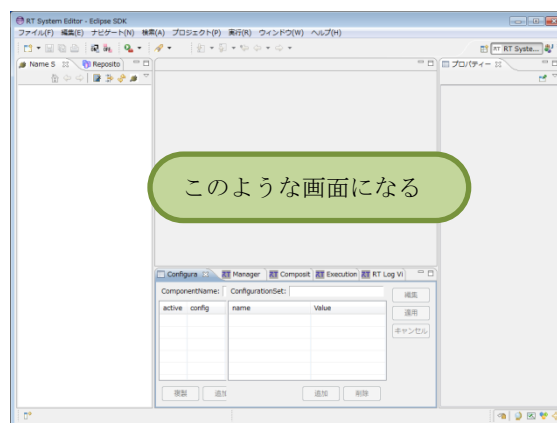
005 その他を選択



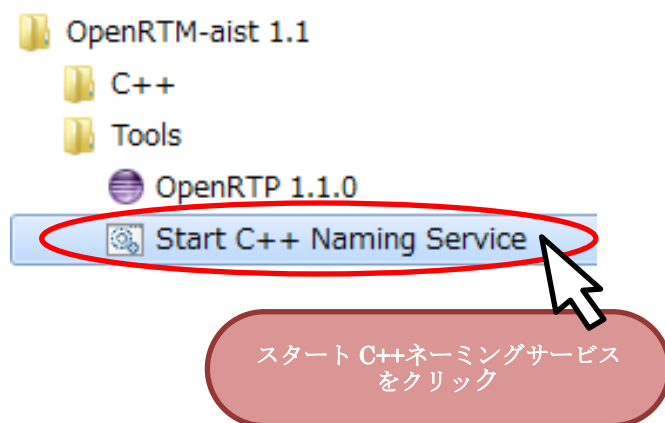
006 RT System Editor を選択. OK



007 RT System Editor の画面



008 NamingService を起動 ②



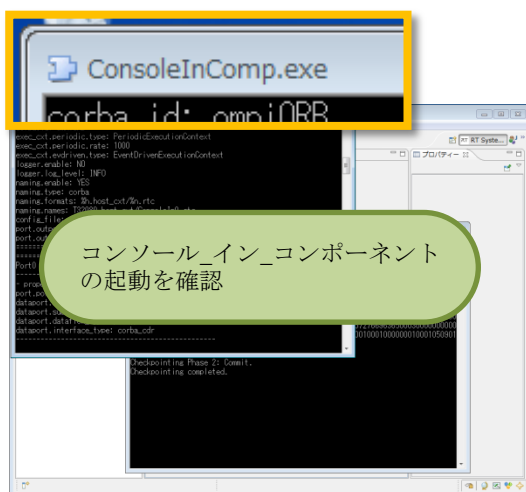
009 NamingService の起動画面



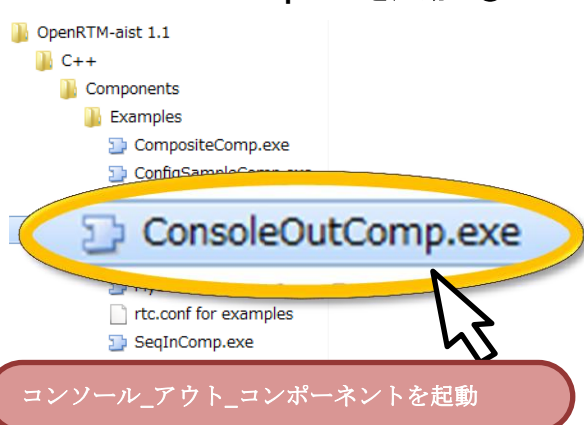
010 ConsoleInComp.exe を起動 ③



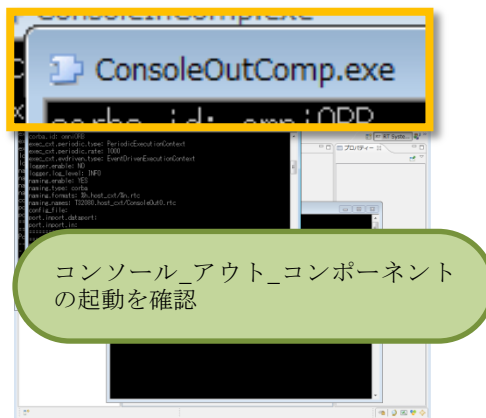
011 ConsoleInComp.exe 起動画面



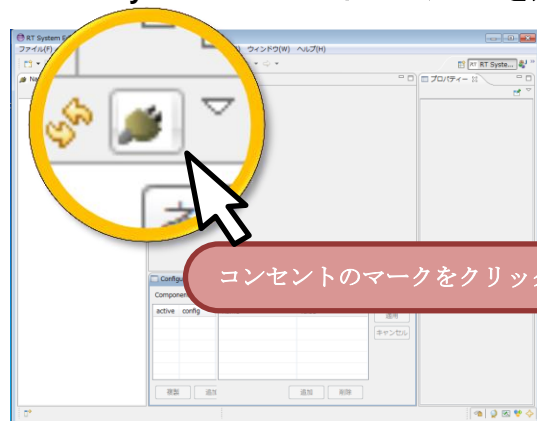
012 ConsoleOutComp.exe を起動 ④



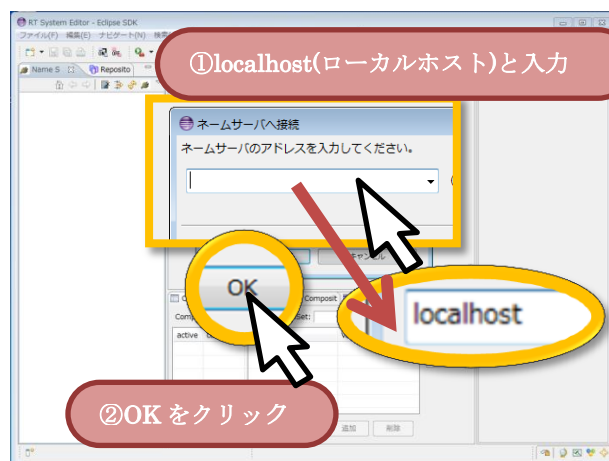
013 ConsoleOutComp.exe 起動画面



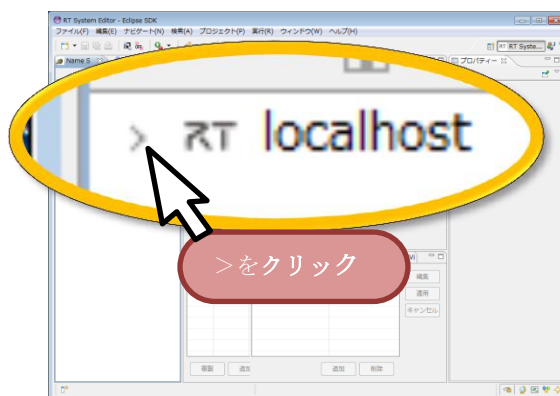
014 RT System Editor でネームサーバを追加



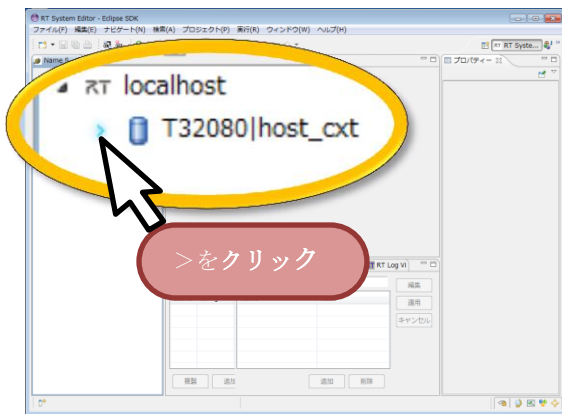
015 ネームサーバのアドレスを入力 localhost と 入力. OK



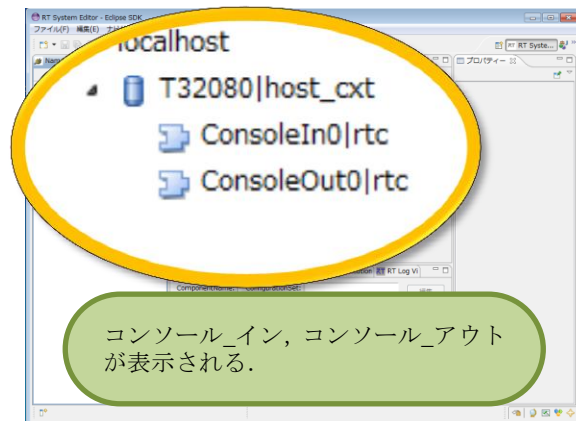
016 localhost の前の「>」クリック



017localhost の下の「>」をクリック。



018 RT コンポーネントが表示される。

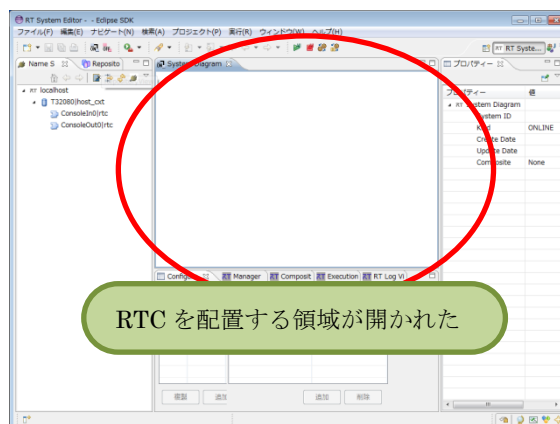


019 新しいシステムエディタを開く
「on」をクリック

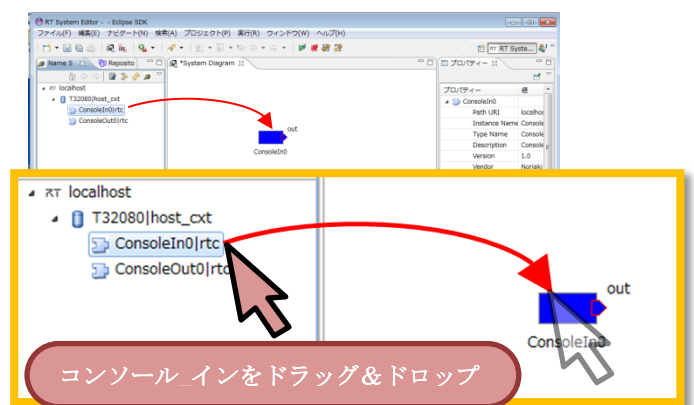


020 System Diagram を確認

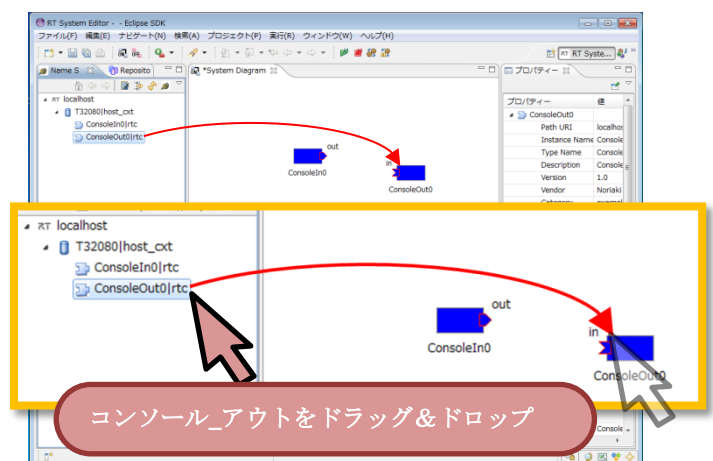
ここに RT コンポーネントを配置する



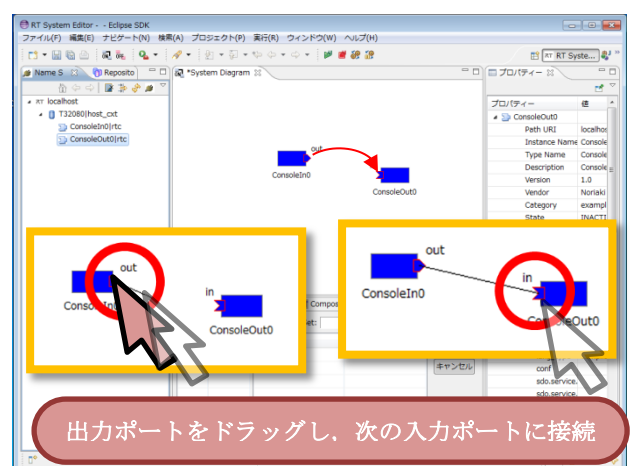
021 ConsoleInComp.exe をドラッグ&ドロップ⑤



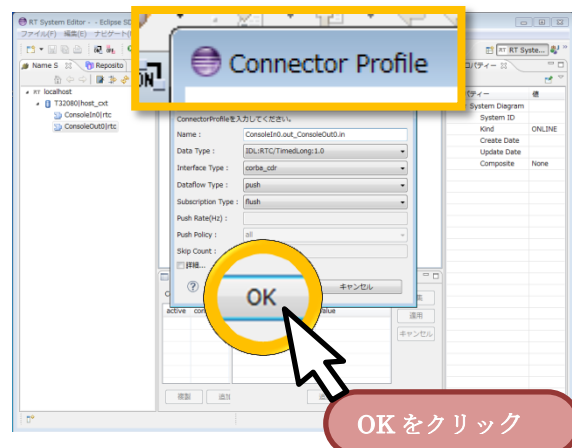
022 ConsoleOutComp.exe をドラッグ&ドロップ⑥



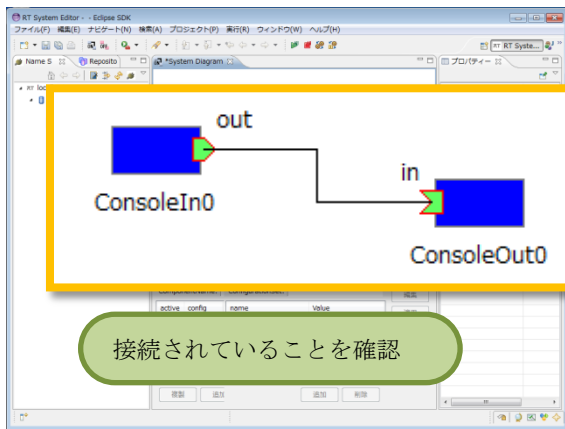
023 コネクター(接続線)をドラッグ&ドロップ ⑦



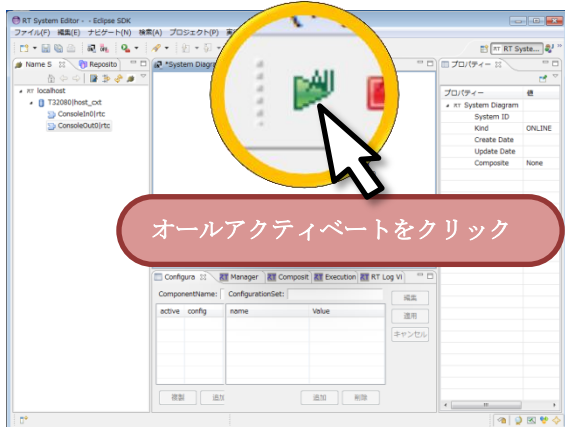
024 コネクタープロファイル→OK



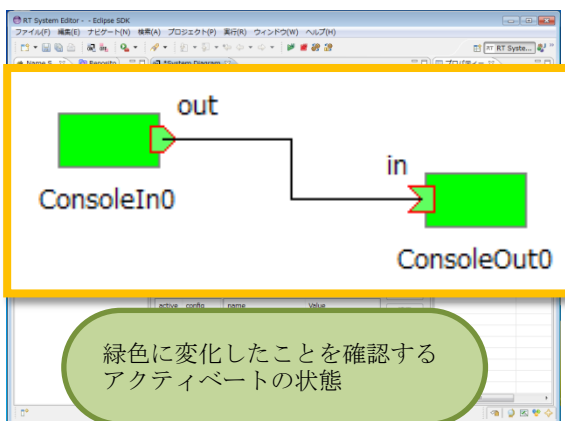
025 接続された状態



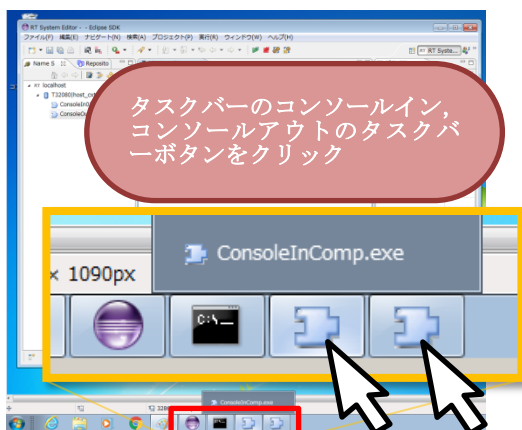
026 コンポーネントをアクティベートする ⑧ 緑色の三角マークを押す



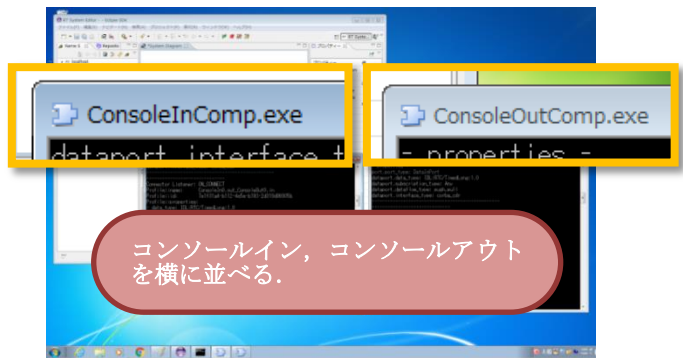
027 アクティベートすると緑色に変わる



028 ConsoleInComp.exe, ConsoleOutComp.exe



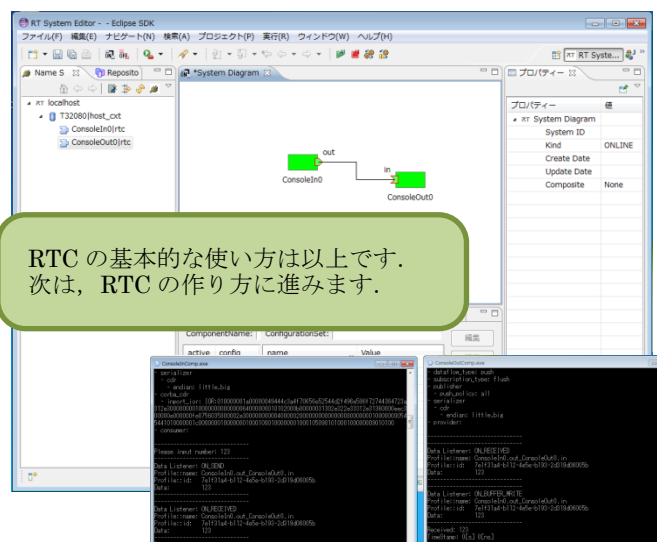
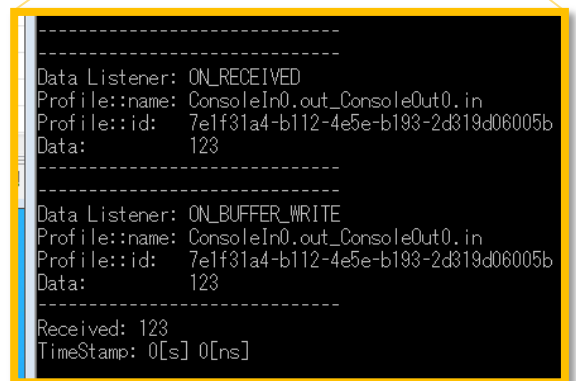
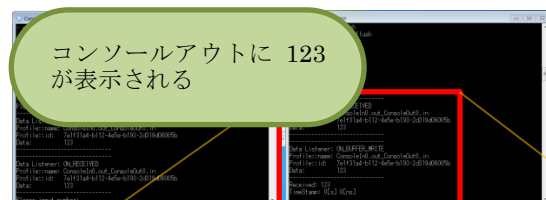
029 横に並べて配置



030 ConsoleInComp.exe に「123」を入力

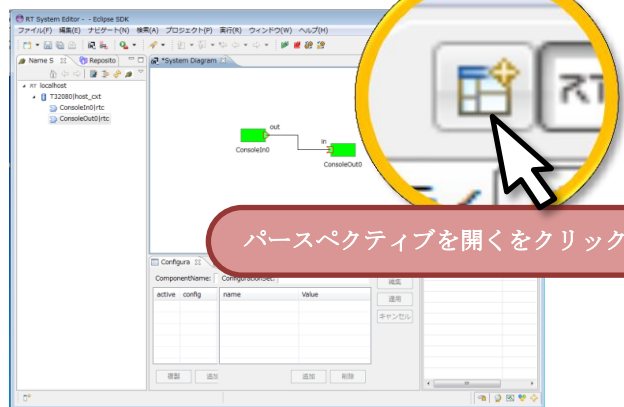


031 ConsoleOutComp.exe に 123 が出力される

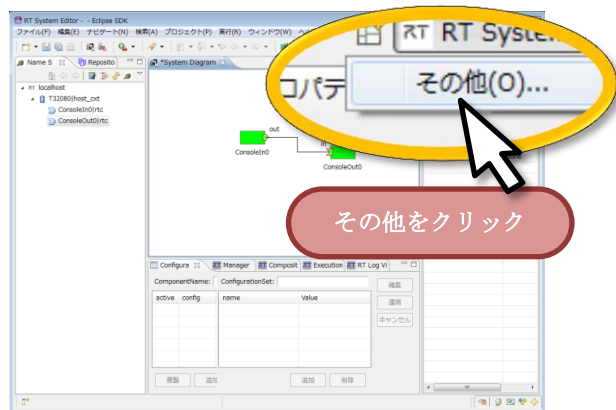


RTC の作り方 (032~067)

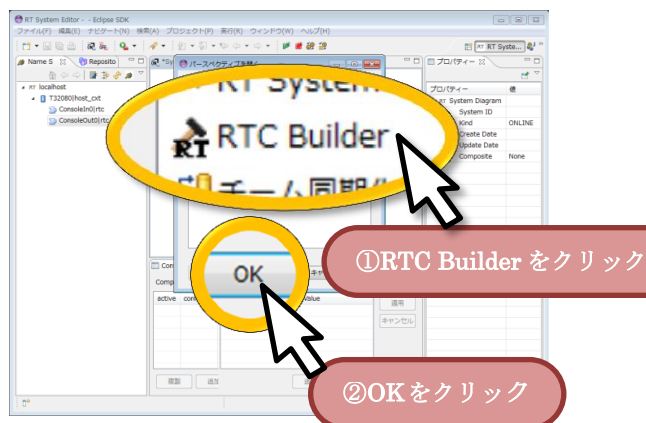
032 RTC ビルダーに切り替える



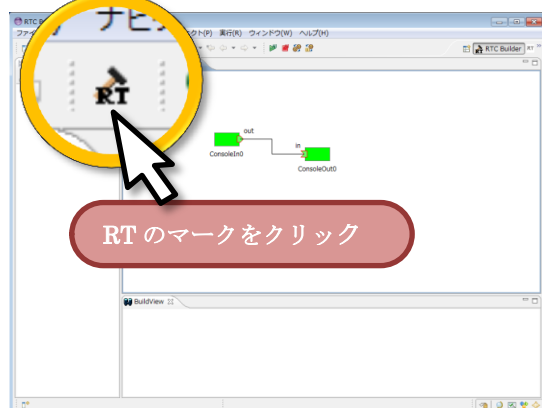
033 その他を開く



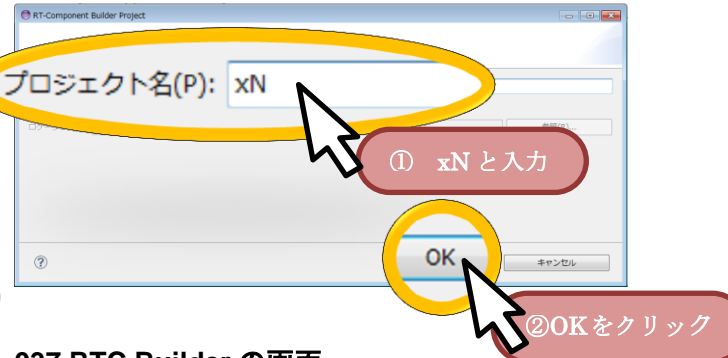
034 RTC Builder を選択し、OK ⑨



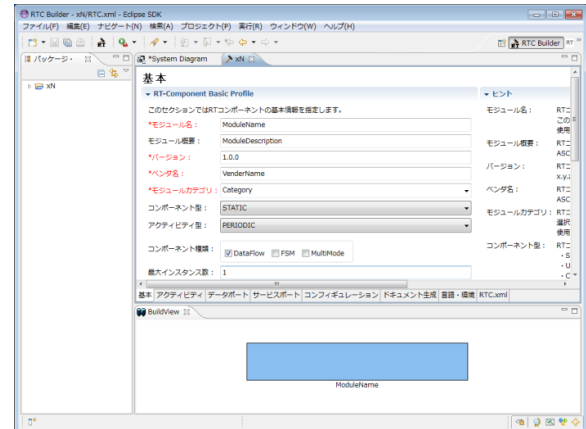
035 RTC Builder を開く



036 プロジェクト名を付ける 「xN」とする。OK.

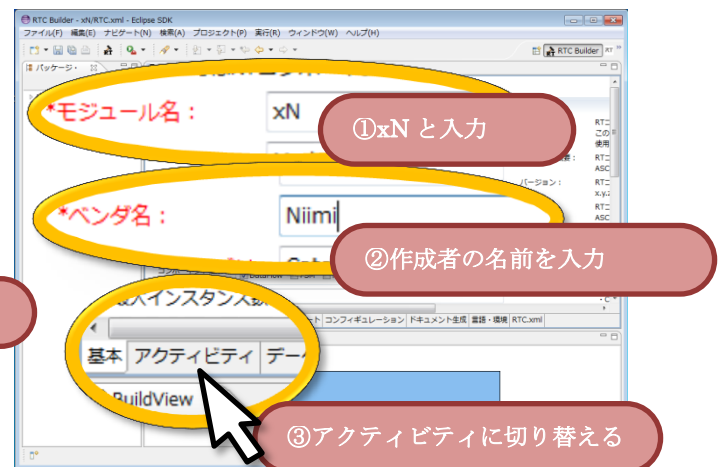


037 RTC Builder の画面

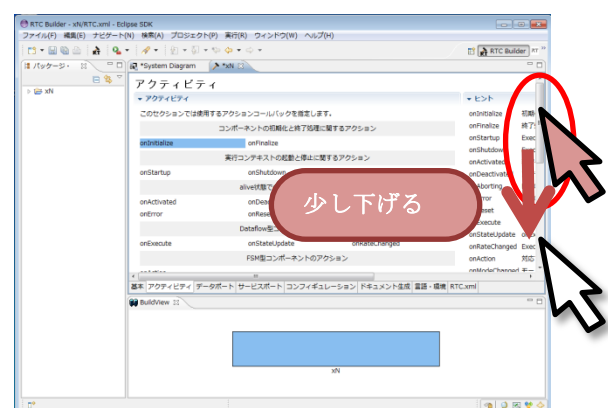


038 モジュール名を xN

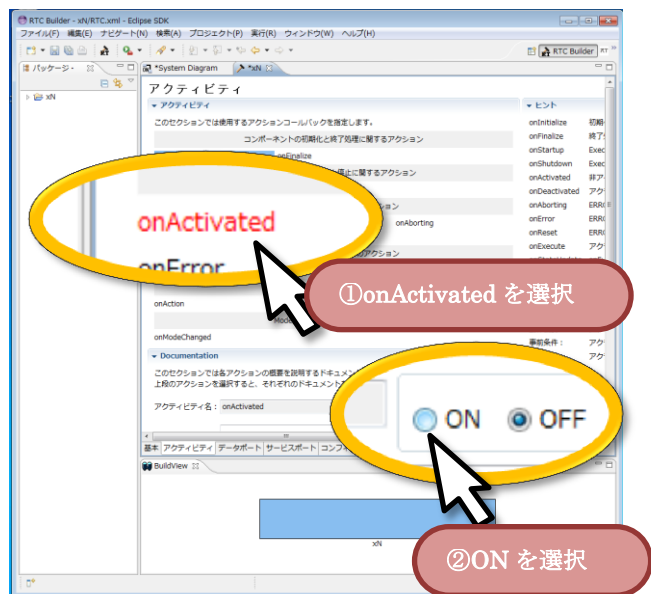
ベンダー名に作成者の名前を入れる
アクティビティに切り替える



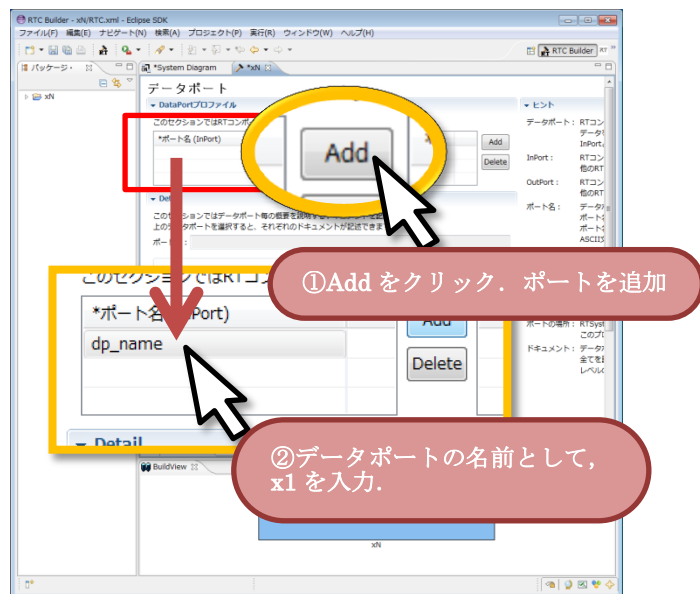
039 アクティビティの設定画面



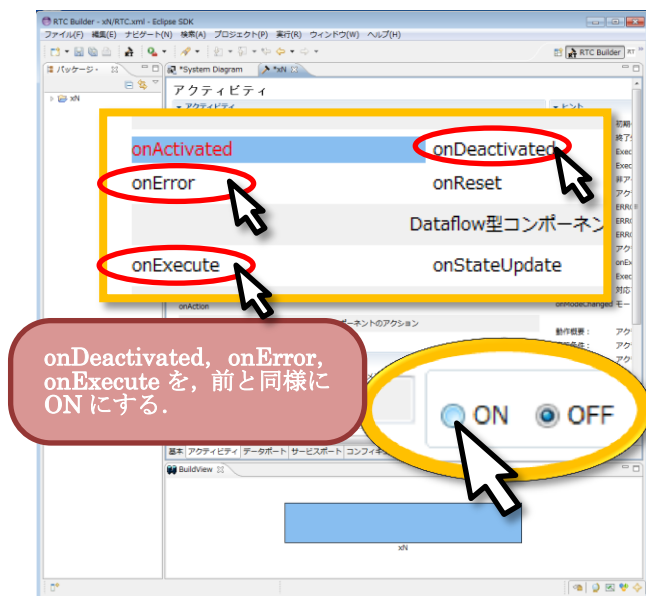
040 onActivated を ON にする



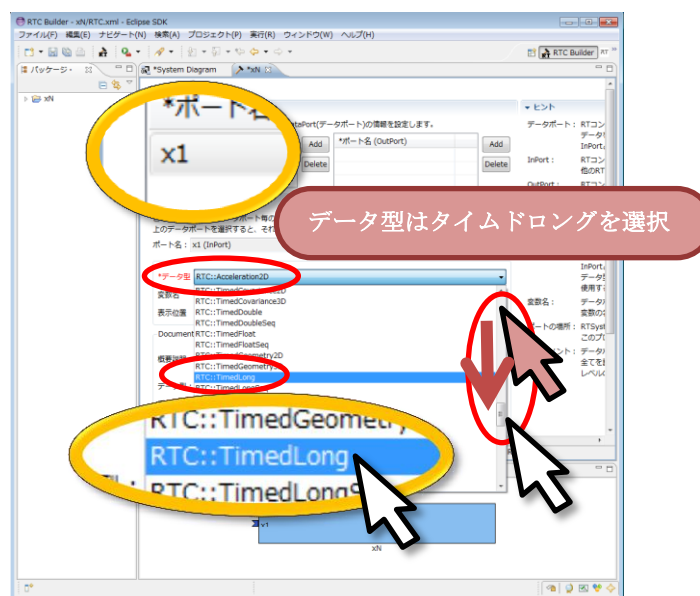
043 データポートの画面. InPort 追加



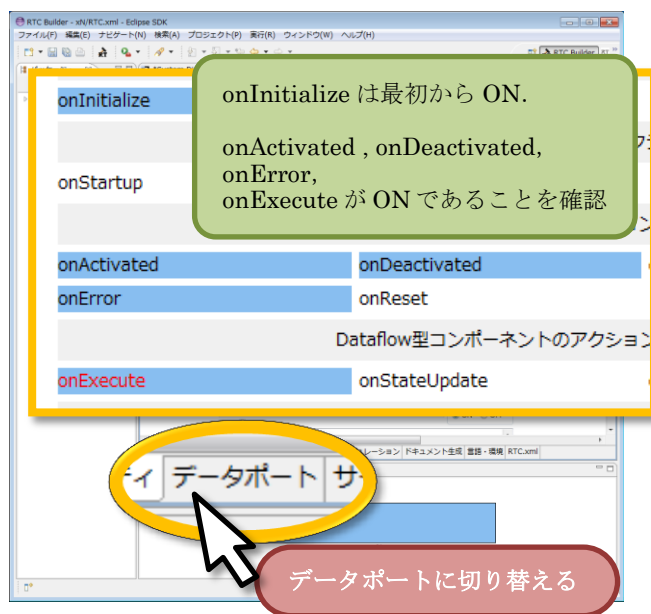
041 onError, onDeactivated, onExecute を ON



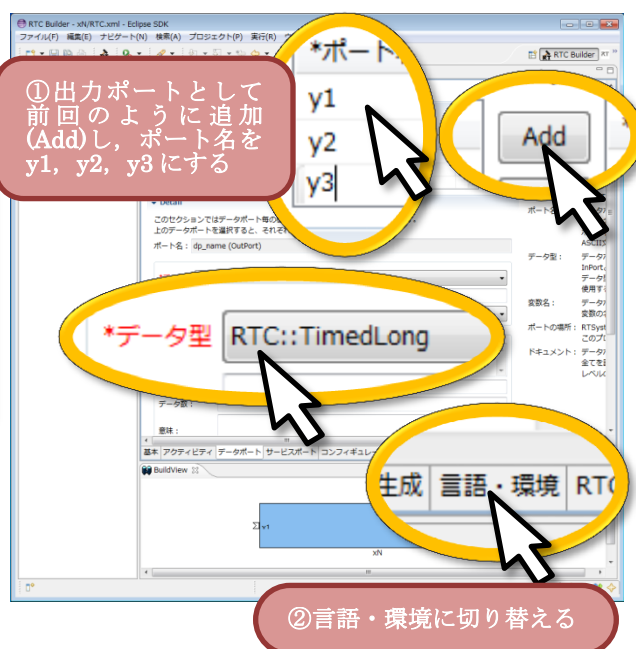
044 データ型.



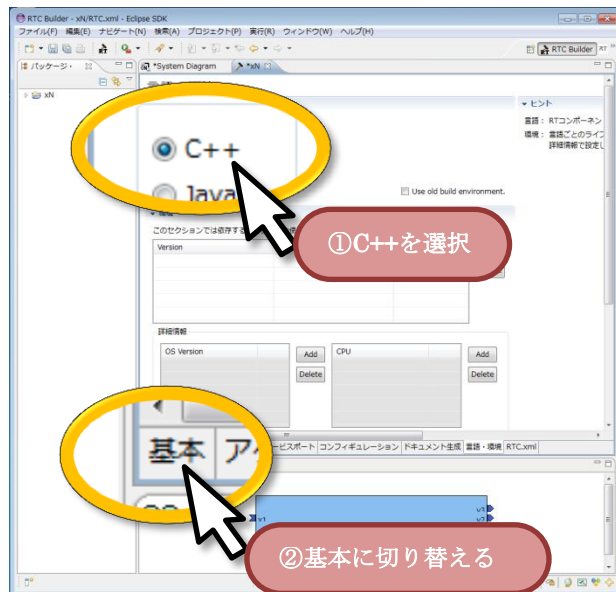
042 5ヶ所が水色になる. データポートに切替え



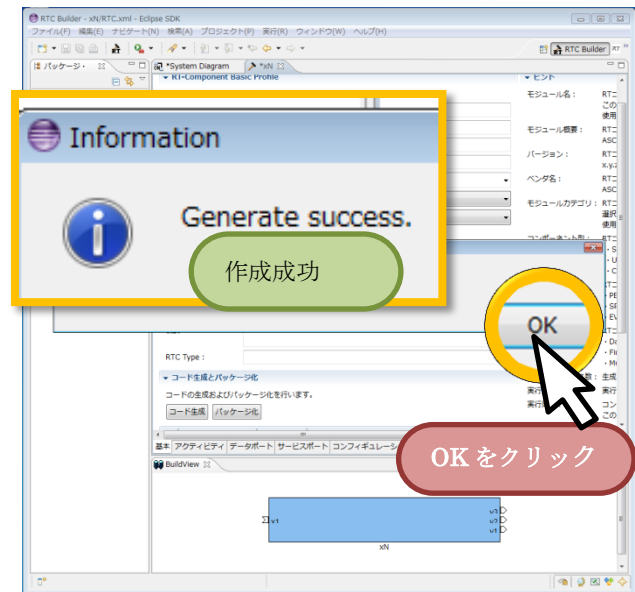
045 OutPort の設定. (y1,y2,y3 [TimedLong])



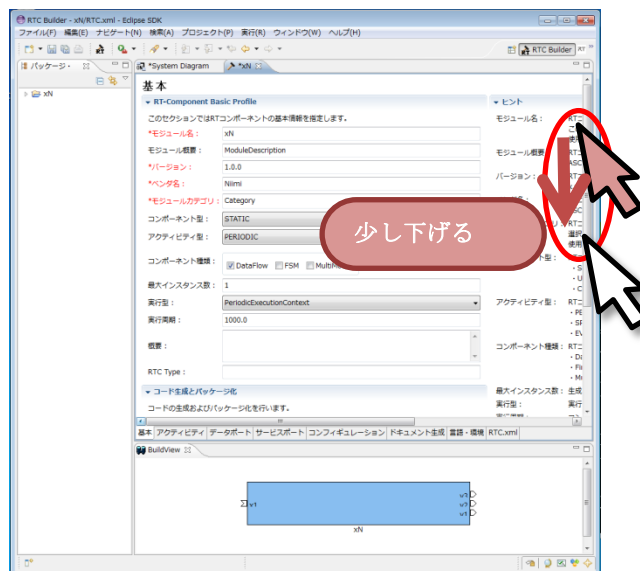
046 C++を設定。基本に戻る。



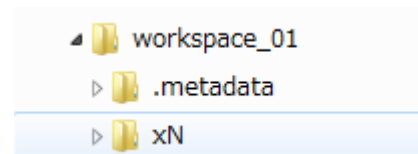
049 成功



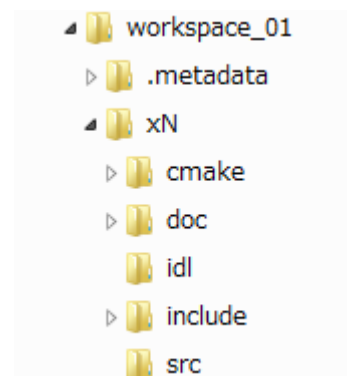
047 基本の画面



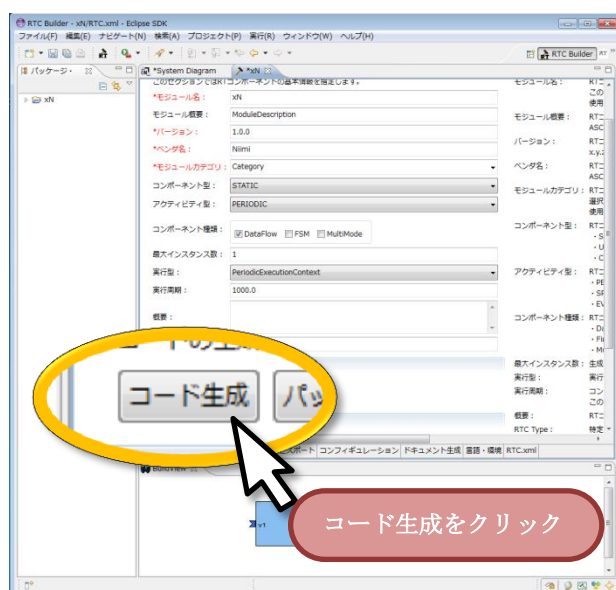
050 エクスプローラーでファイルを確認する
[002]で、C:\workspace_01 を設定した。
[036]で、プロジェクト名 xN を設定した。



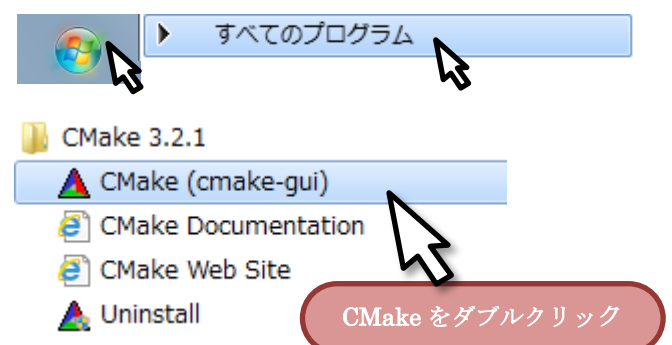
051 xN の下のフォルダーを確認する。
build というフォルダーは、今はない。
CMake で作成。



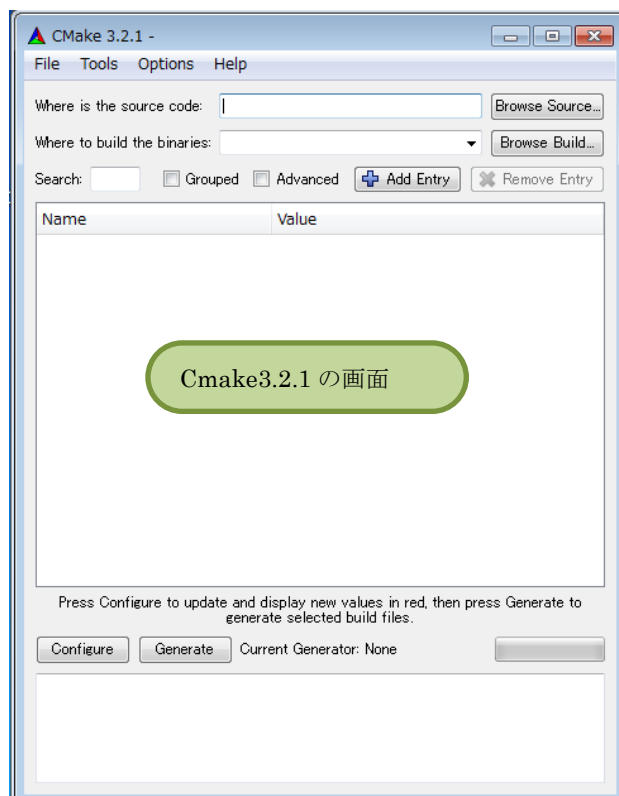
048 基本の画面



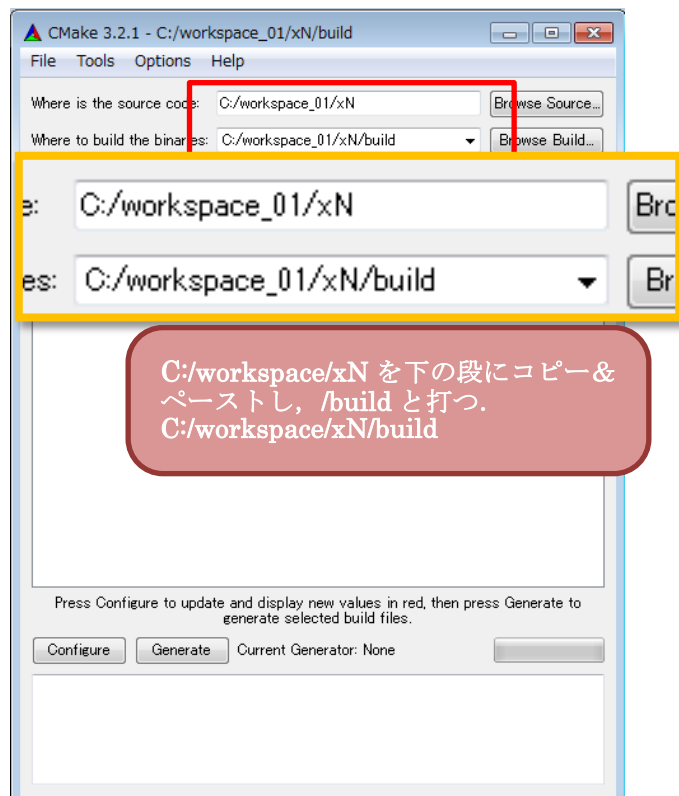
052 Cmake3.2.1 を起動する。⑩



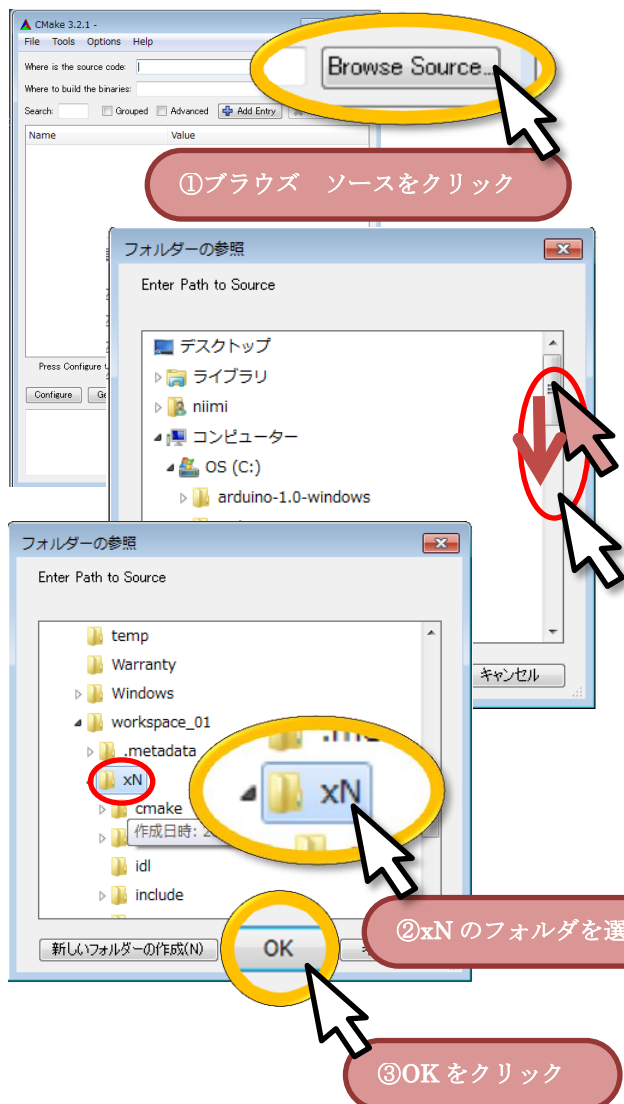
052 Cmake3.2.1 の画面



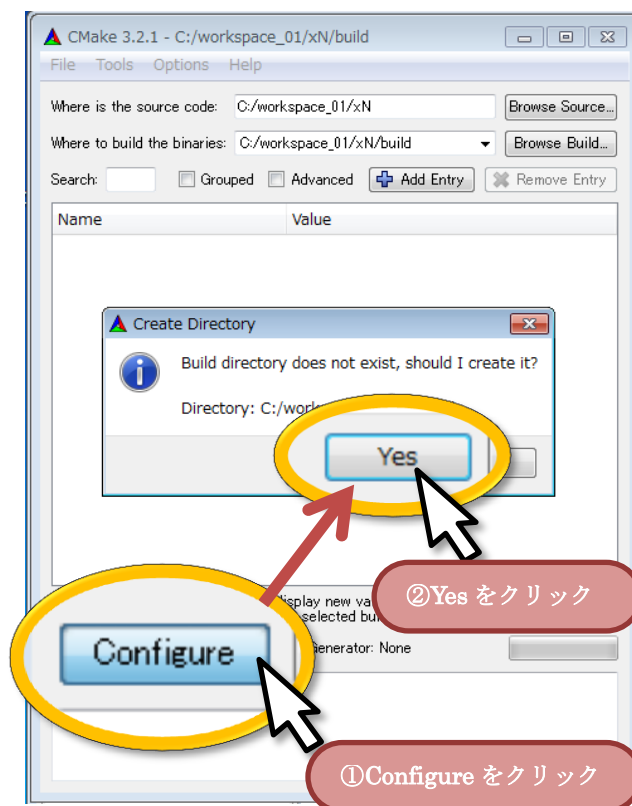
054 build の場所を指定する.



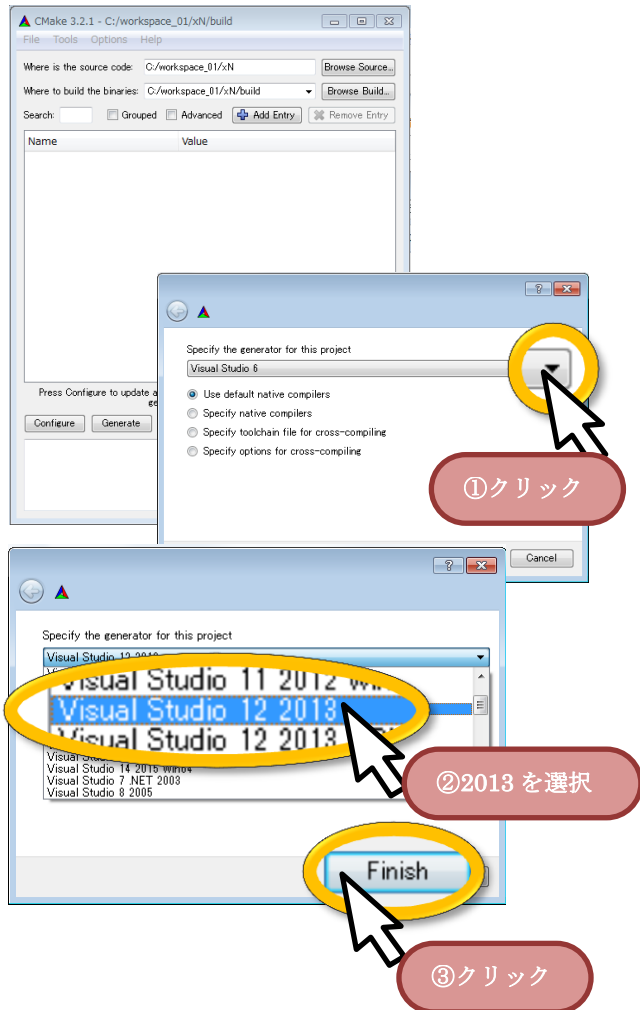
053 ソースコードの場所を指定する.



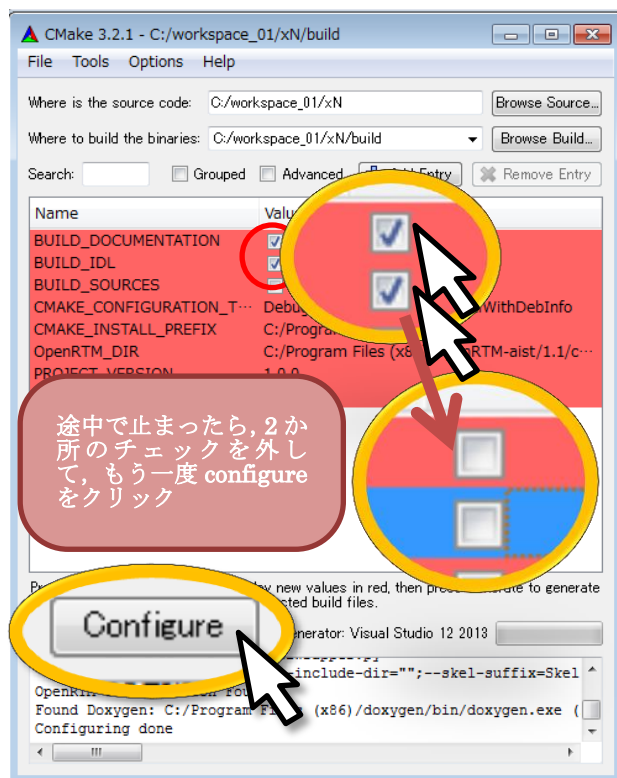
055 configure をクリック. Build というディレクトリ作成の確認画面が出る. Yes.



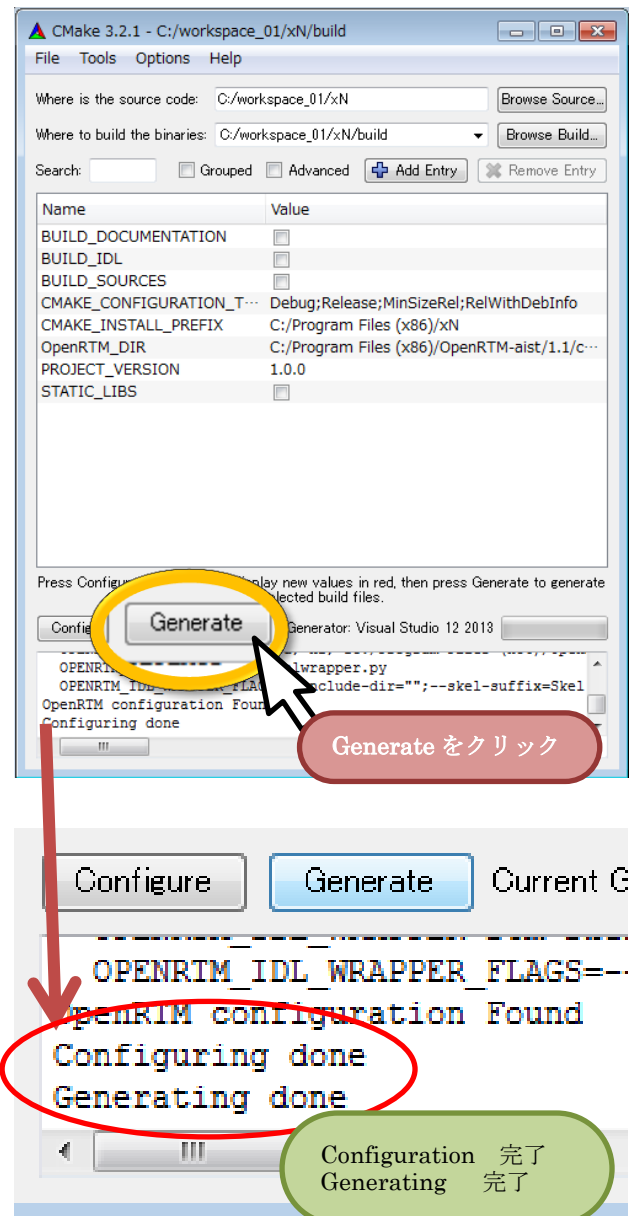
056 Visual Studio のバージョンを選択する。
ここでは、Visual Studio 12 2013 を使用した。



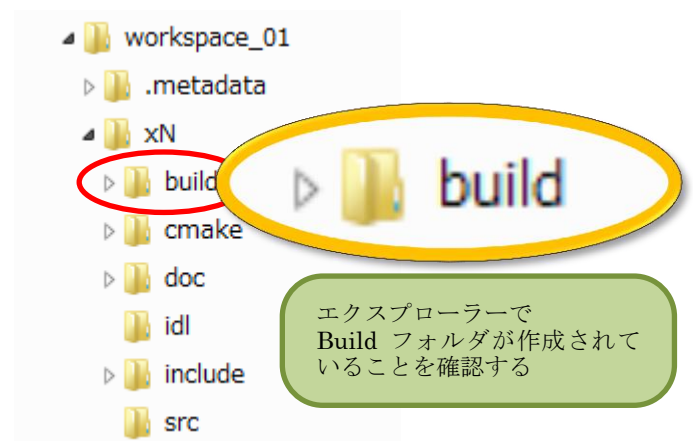
057 途中で止まったら、2 か所のチェックを外して、もう一度 configure をクリック



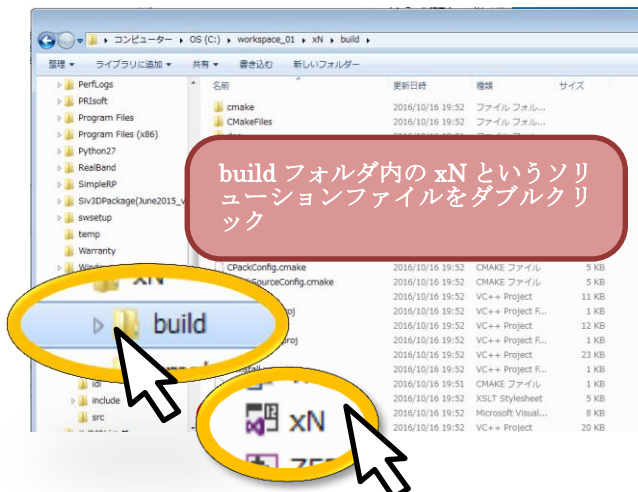
058 Generate をクリック



059 エクスプローラーで xN の下に「build」のフォルダーが作成していることを確認する



060 エクスプローラーで xN の下の build を開く xN のソリューションファイルをダブルクリック

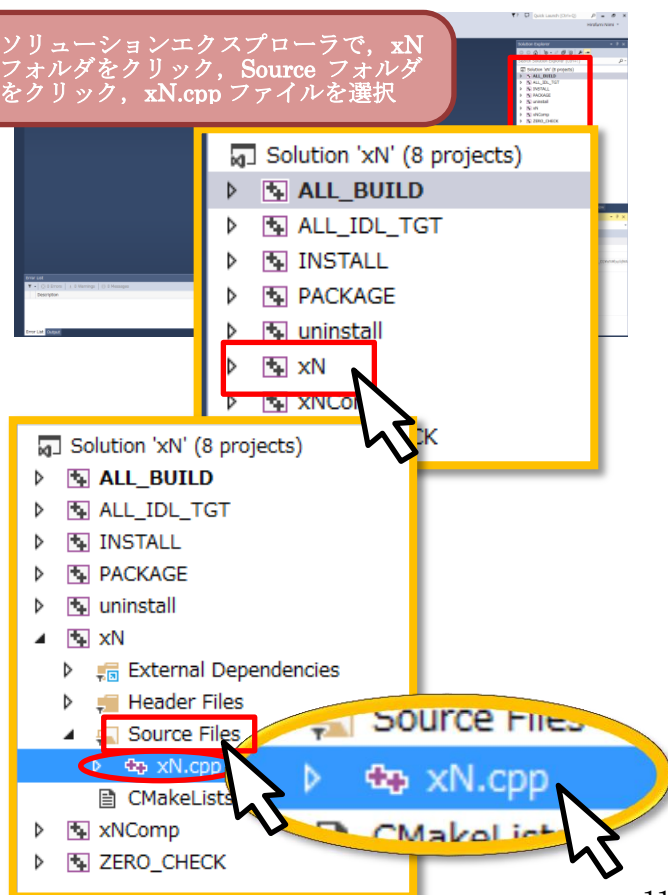


061 Visual Studio の起動中の画面 ①

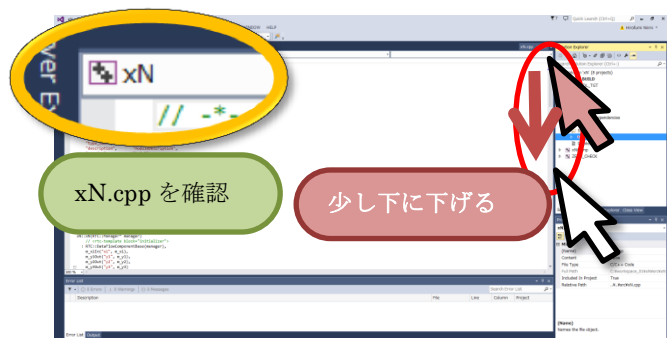


061 Visual Studio の画面

ソリューションエクスプローラーで、xN フォルダをクリック、Source フォルダをクリック、xN.cpp ファイルを選択



061 xN のソースファイル



062 onExecute にコードを書く準備

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

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

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

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

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

ここにコードを書く

063 onExecute にコードを書く(2倍,10倍,100倍)

P14に大きく表示しています。

最初、2倍のみ入力。

```
RTC::ReturnCode_t xN::onExecute(RTC::UniqueId ec_id)
{
    if (m_x1In.isNew()) {
        m_x1In.read();
        m_y1.data = m_x1.data * 2;

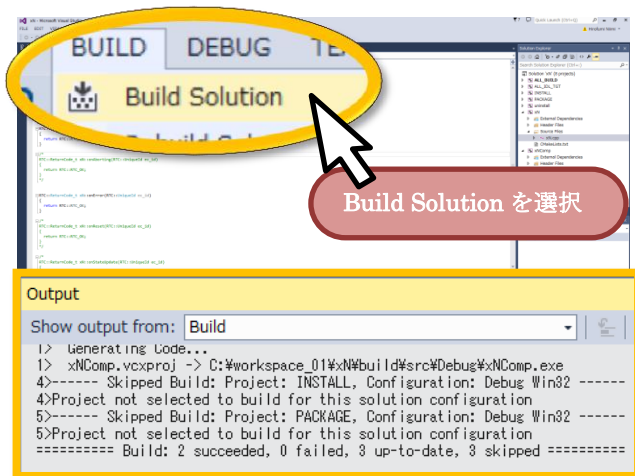
        m_y1Out.write();

        std::cout << m_y1.data << "=" << m_x1.data << "*" 2" << std::endl;
    }
    return RTC::RTC_OK;
}
```

コピーで10倍、100倍を作る

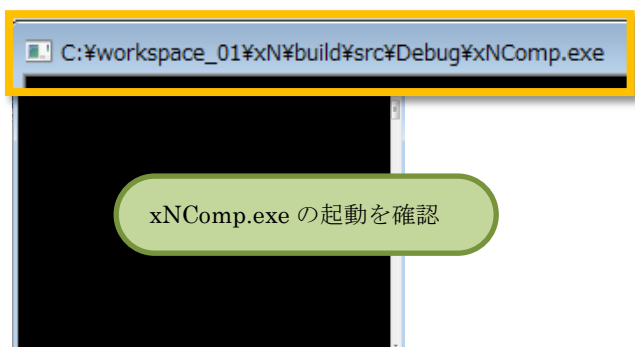
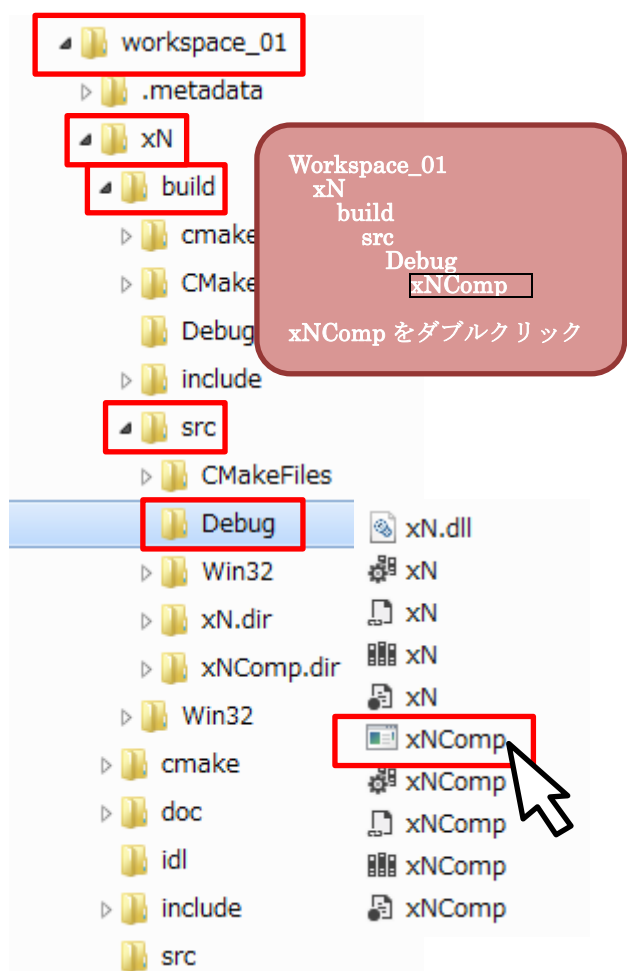
```
RTC::ReturnCode_t xN::onExecute(RTC::UniqueId ec_id)
{
    if (m_x1In.isNew()) {
        m_x1In.read();
        m_y1.data = m_x1.data * 2;
        m_y2.data = m_x1.data * 10;
        m_y3.data = m_x1.data * 100;
        m_y1Out.write();
        m_y2Out.write();
        m_y3Out.write();
        std::cout << m_y1.data << "=" << m_x1.data << "*" 2" << std::endl;
        std::cout << m_y2.data << "=" << m_x1.data << "*" 10" << std::endl;
        std::cout << m_y3.data << "=" << m_x1.data << "*" 100" << std::endl;
    }
    return RTC::RTC_OK;
}
```

064 Build

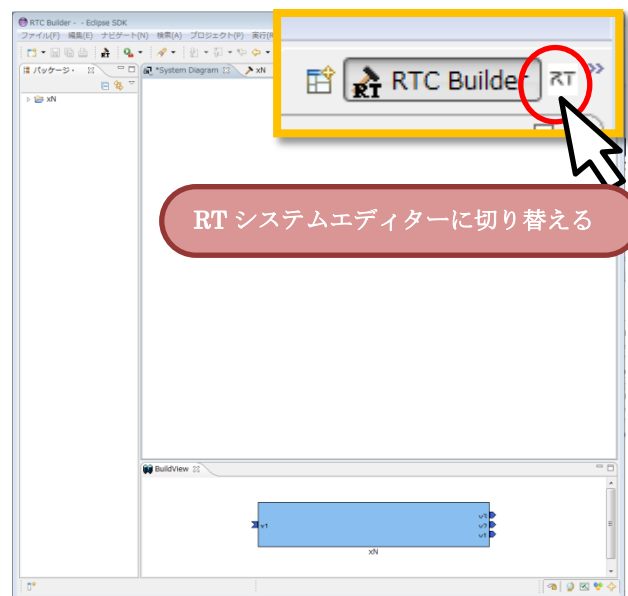


Build succeeded(ビルド成功)を確認

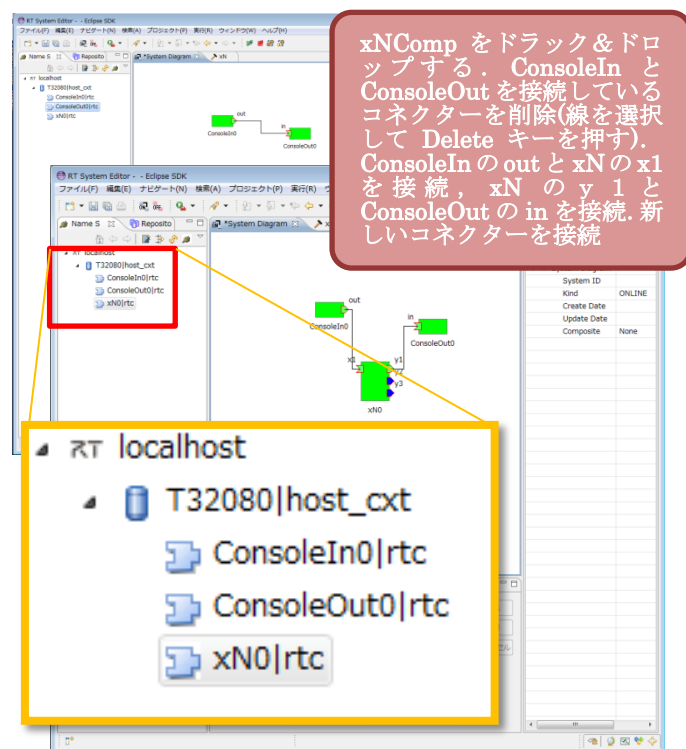
064 xNComp を起動 ⑫



065 RT システムエディターに切り替える

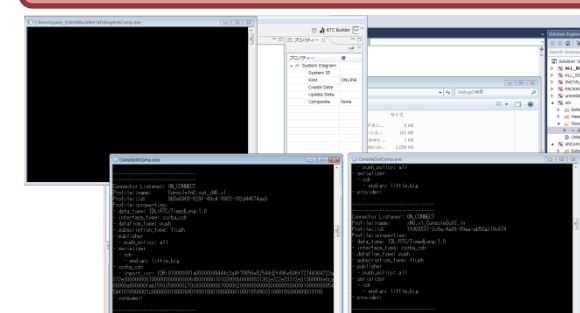


066 xNComp をドラッグ＆ドロップする. 前のコネクタを削除「選択して Delete」. 新しいコネクタを接続



067 動作確認

ConsoleInComp.exe に「123」を入力すると, ConsoleOutComp.exe に 246 が出力される



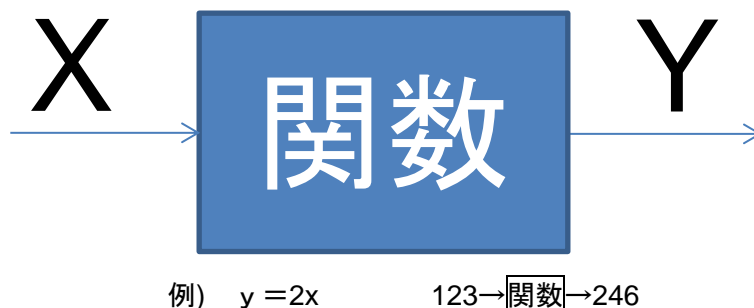
解説

RTC とは？

数学の時間に習った関数のようなものです。

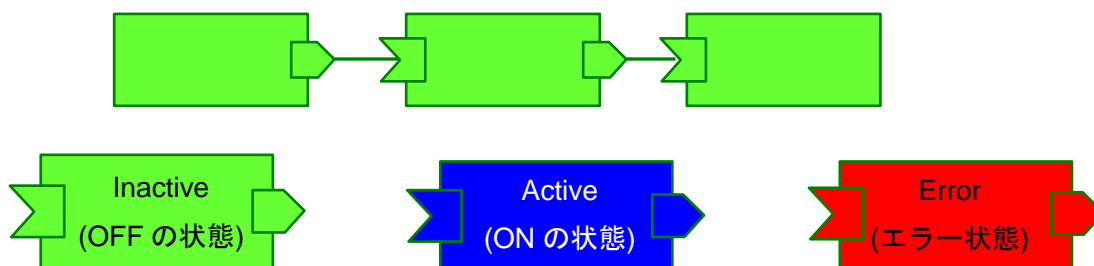
数字を入力すると、一定のルールに基づき、値が出力されます。

関数の部分は、目的の機能を実現するためプログラムに相当します。



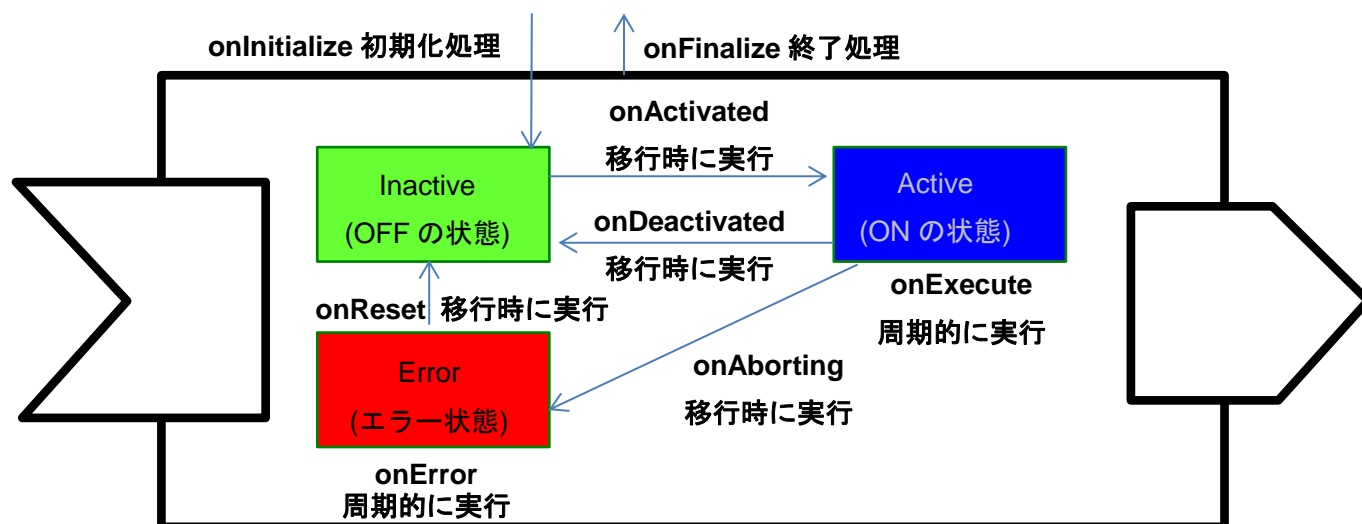
RTC は何が便利なの？

RTC では、入力データの型、出力データの型、内部の状態などのルールが決められています。入力データの型、出力データの型のルールが決められているため、同じ型のコネクタを接続して利用することができます。また、内部状態は、Inactive(OFF の状態)、Active(ON の状態)、Error(エラー状態)が、青、緑、赤の色で示される為、RTC の状態を監視することができます。

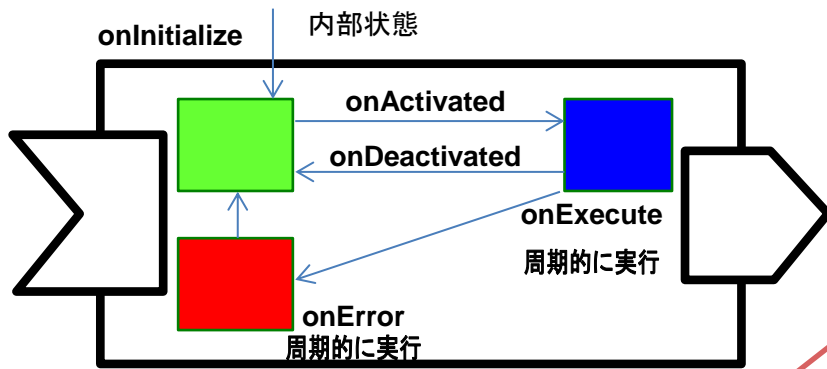


内部状態とは？

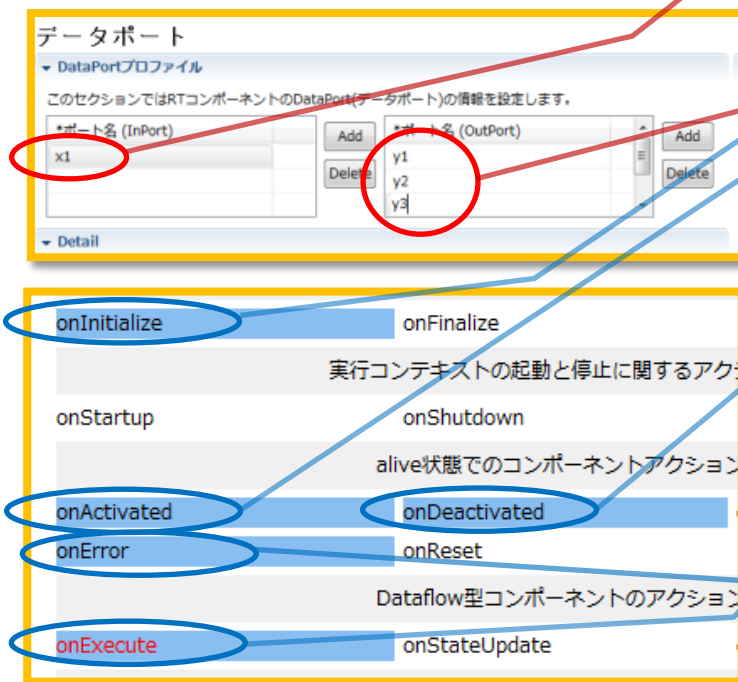
RTC は、「Inactive(緑)」、「Active(青)」、「Error(赤)」の内部状態を持つ。RT システムエディター上に配置した最初は、「Inactive 状態」である。RTC をアクティベート(スイッチ ON)すると、「Active 状態」になる。RTC をディアクティベート(スイッチ OFF)すると、「Inactive 状態」になる。RTC にトラブルが発生すると、「Error 状態」になる。「Error 状態」から「Inactive 状態」にリセットする必要がある。「Error 状態」から「Active 状態」に直接移行することはできない。



練習課題の 内部状態, RT システムエディターを使用したポートの設定, xN のソースファイルの関係



RT システムエディターを使用したポートの設定



xN のソースファイル

```

RTC::ReturnCode_t xN::onInitialize()
{
    // Registration: InPort/OutPort/Service
    // <rtc-template block="registration">
    // Set InPort buffers
    addInPort("x1", m_x1In);

    // Set OutPort buffer
    addOutPort("y1", m_y1Out);
    addOutPort("y2", m_y2Out);
    addOutPort("y3", m_y3Out);
}

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

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

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

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

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

練習課題の メインプログラム(onExecute のコード)

データポートで指定したポート名は, (x1, y1, y2, y3)である. データは, (m_x1.data, m_y1.data, m_y2.data, m_y3.data)という変数名で扱う. データを読み書きする場合は, (m_x1In.read(), m_y1Out.write(), m_y2Out.write(), m_y3Out.write())を使う. m_x1 と m_x1In の違いに注意が必要.

「std::cout<<m_x1.data<<"*2"<<std::endl」は, コンソールに表示するための標準的な書き方.

```

RTC::ReturnCode_t xN::onExecute(RTC::UniqueId ec_id)
{
    if (m_x1In.isNew()) {
        m_x1In.read();
        m_y1.data = m_x1.data * 2;
        m_y2.data = m_x1.data * 10;
        m_y3.data = m_x1.data * 100;
        m_y1Out.write();
        m_y2Out.write();
        m_y3Out.write();
        std::cout << m_y1.data << "=" << m_x1.data << "* 2" << std::endl;
        std::cout << m_y2.data << "=" << m_x1.data << "* 10" << std::endl;
        std::cout << m_y3.data << "=" << m_x1.data << "* 100" << std::endl;
    }
    return RTC::RTC_OK;
}
    
```