

コンポーネント作成

会津大学 R T ミドルウェア講習会

目次

1	はじめに.....	1
1.1	OpenCV とは.....	1
1.2	作成する RT コンポーネント	1
2	cvFlip 関数の RT コンポーネント化	1
2.1	cvFlip 関数について	1
2.2	コンポーネントの仕様	2
2.2.1	名称.....	2
2.2.2	コンポーネントの動作	2
2.2.3	仕様まとめ	3
3	Flip コンポーネントの雛型を生成する	4
3.1	RTCBuilder を起動する	4
3.2	新規プロジェクトを作成する.....	6
3.3	RTC プロファイルエディタを起動する	8
3.4	プロファイルの情報を入力しコードを生成する	9
3.4.1	基本.....	9
3.4.2	アクティビティ	10
3.4.3	データポート	11
3.4.4	コンフィギュレーション	12
3.4.5	言語・環境.....	13
3.4.6	コード生成	14
3.5	仮ビルドする	15
3.5.1	CMake を起動する	15
3.5.2	値を指定する	15
3.5.3	プロジェクト種類を指定する.....	16
3.5.4	Configure と Generate	16

3.5.5	Visual Studio を起動する.....	16
3.5.6	ALL_BUILD を実行する.....	17
4	ヘッダ、ソースを編集する.....	17
4.1	アクティビティ処理を実装する.....	17
4.2	ヘッダファイル (Flip.h)を編集する.....	18
4.3	ソースファイル (Flip.cpp)を編集する	19
4.3.1	onActivated()	19
4.3.2	onDeactivated()	19
4.3.3	onExecute()	20
4.4	CMake でビルドに必要なファイルを生成する	21
4.5	VC++でビルドする	22
4.6	コンポーネントを起動する.....	22
4.6.1	NameService を起動する	22
4.6.2	Flip コンポーネントを起動する	23
4.6.3	カメラコンポーネントとビューアコンポーネントを起動する	23
4.6.4	コンポーネントを接続する	23
4.6.5	コンポーネントをアクティブ状態にする	26
4.6.6	動作確認	27
4.6.7	Flip コンポーネントの動作確認	28
5	Flip コンポーネントの全ソース.....	29
5.1	Flip コンポーネントソースファイル (Flip.cpp).....	29
5.2	Flip コンポーネントのヘッダファイル (Flip.h).....	29
5.3	Flip コンポーネントの全ソースコード.....	29

この講習会テキストは下記ページを参考にしています。

- ・チュートリアル（画像処理コンポーネントの作成 Windows 編）

<http://www.openrtm.org/openrtm/ja/node/5022>（2016/1/8 アクセス）

※ 文中の「x.y」や「x.y.z」の表記は使用環境の OpenRTM-aist のバージョンに読み替えてください。

1 はじめに

1.1 OpenCV とは

画像処理・画像解析および機械学習等の機能を持つ C/C++、Java、Python、MATLAB 用ライブラリです。

1.2 作成する RT コンポーネント

Flip コンポーネント: OpenCV ライブラリが提供する様々な画像処理関数のうち、`cvFlip()` 関数を用いて画像の反転を行う RT コンポーネント

2 cvFlip 関数の RT コンポーネント化

OpenCV の `cvFlip` 関数を使用して、入力された画像を左右または上下に反転して出力するコンポーネントを作成します。

作成手順としては

- 1)コンポーネントの仕様を決定
- 2)RTCBuilder を用いたソースコードのひな形の作成
- 3)アクティビティ処理の実装
- 4)コンポーネントの動作確認

になります。

2.1 cvFlip 関数について

`cvFlip` 関数は、OpenCV で標準的に用いられている関数です。入力された画像データを反転させて出力する機能があります。反転させる軸は垂直軸、水平軸、両軸と三種類あり引数で設定することができます。

2.2 コンポーネントの仕様

2.2.1 名称

これから作成するコンポーネントを **Flip** コンポーネントという名称にします。

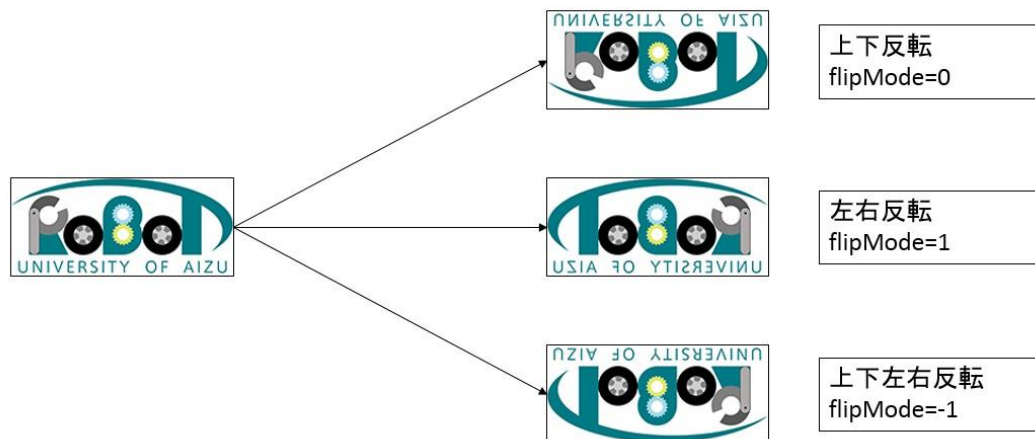
2.2.2 コンポーネントの動作

このコンポーネントの動作としては画像データを入力ポート(**InPort**)から受け取り反転処理した画像データを出力ポート(**OutPort**)へ出力します。それぞれのポートの名前を入力ポート(**InPort**)名:**originalImage**,出力ポート(**OutPort**)名:**flippedImage** とします。

これらのコンポーネントのデータポートは画像の入出力に **CameraImage** 型を使用します。

また、画像を反転させる方向は、左右反転、上下反転、上下左右反転の 3 通りです。実行時に反転方向を指定できるように、**RT** コンポーネントのコンフィギュレーション機能を使用します。パラメータ名は **flipMode** という名前にします。

flipMode は **cvFlip** 関数の仕様に合わせて、型は **int** 型とし上下反転、左右反転、上下左右反転にそれぞれに 0, 1, -1 を割り当てることにします。



コンポーネント作成

2.2.3 仕様まとめ

以上から Flip コンポーネントの仕様をまとめると下記のようになります。

コンポーネント名称	Flip
InPort	
ポート名	originalImage
型	CameraImage
意味	入力画像
OutPort	
ポート名	flippedImage
型	CameraImage
意味	反転された画像
Configuration	
パラメータ名	flipMode
型	int
意味	反転モード 上下反転: 0 左右反転: 1 上下左右反転: -1

3 Flip コンポーネントの雛型を生成する

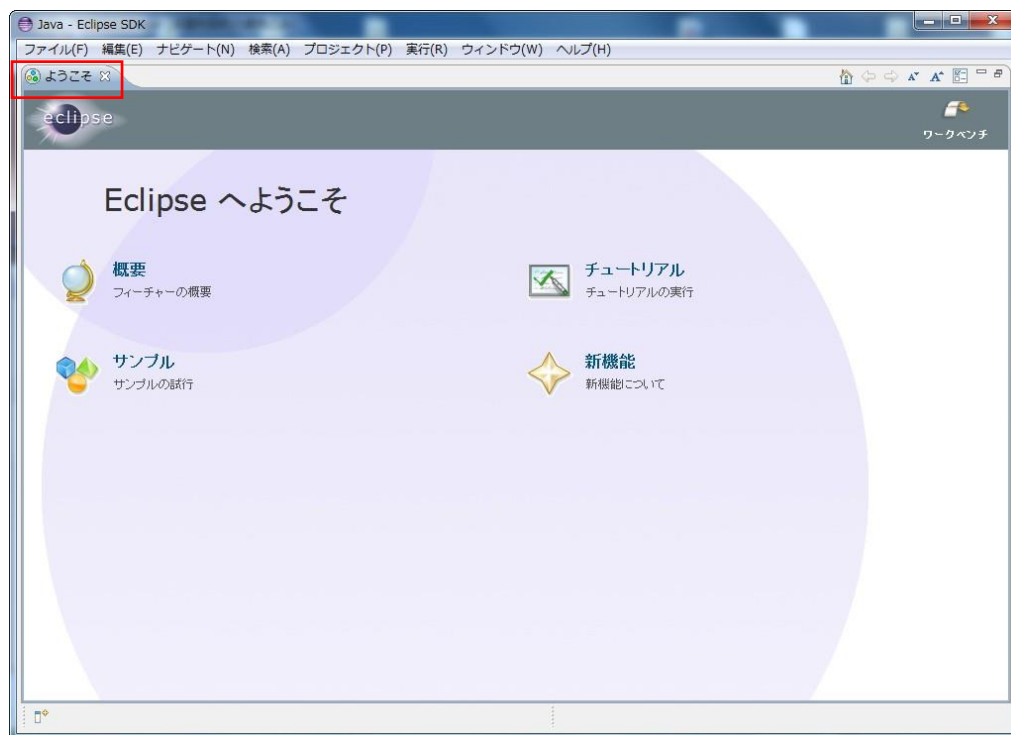
Flip コンポーネントの雛型の生成方法を説明します。

3.1 RTCBuilder を起動する

OpenRTP を起動させると作成物を保存するディレクトリを指定します。ここでは C ドライブ直下の下記ディレクトリに保存します。

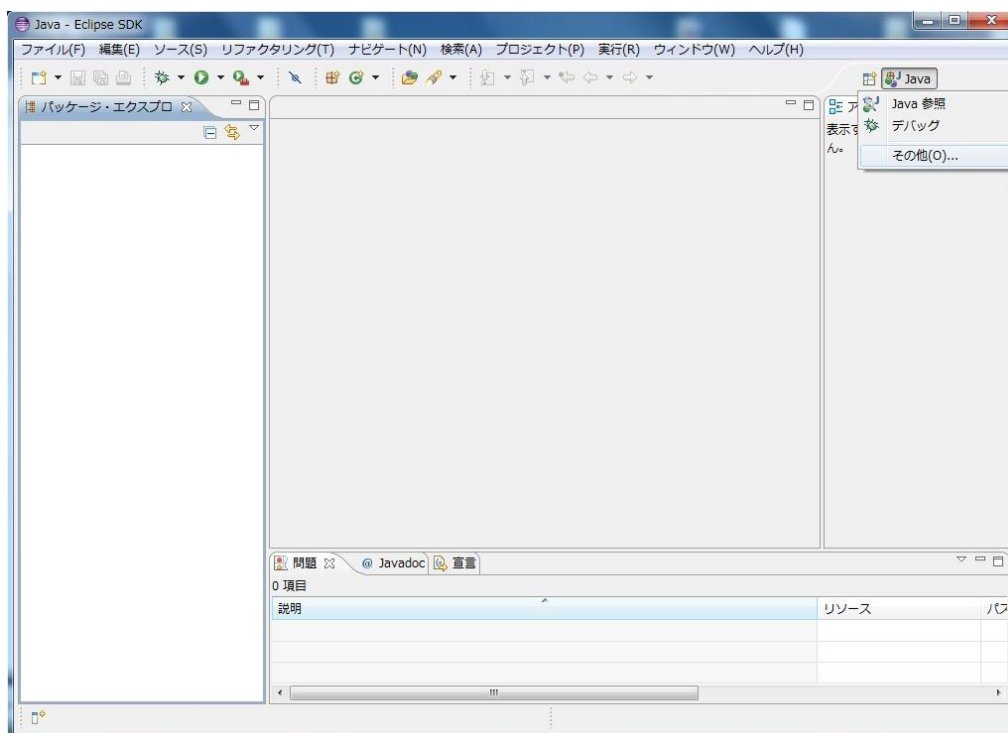
[C:\rtcws]

最初に起動したときに下記画面が表示されます。この画面は使用しないので左上の×ボタンを押します。

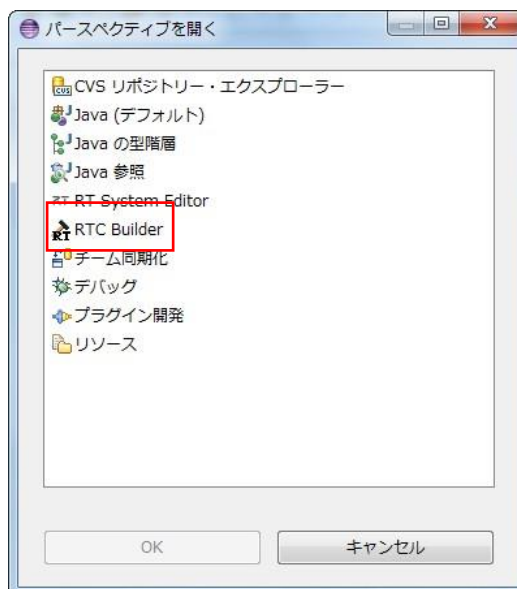


コンポーネント作成

×ボタンを押すと下記画面が表示されます。右上の[その他]をクリックしてください。



下記ウィンドウが表示されるので[RTC Builder]を選択します。

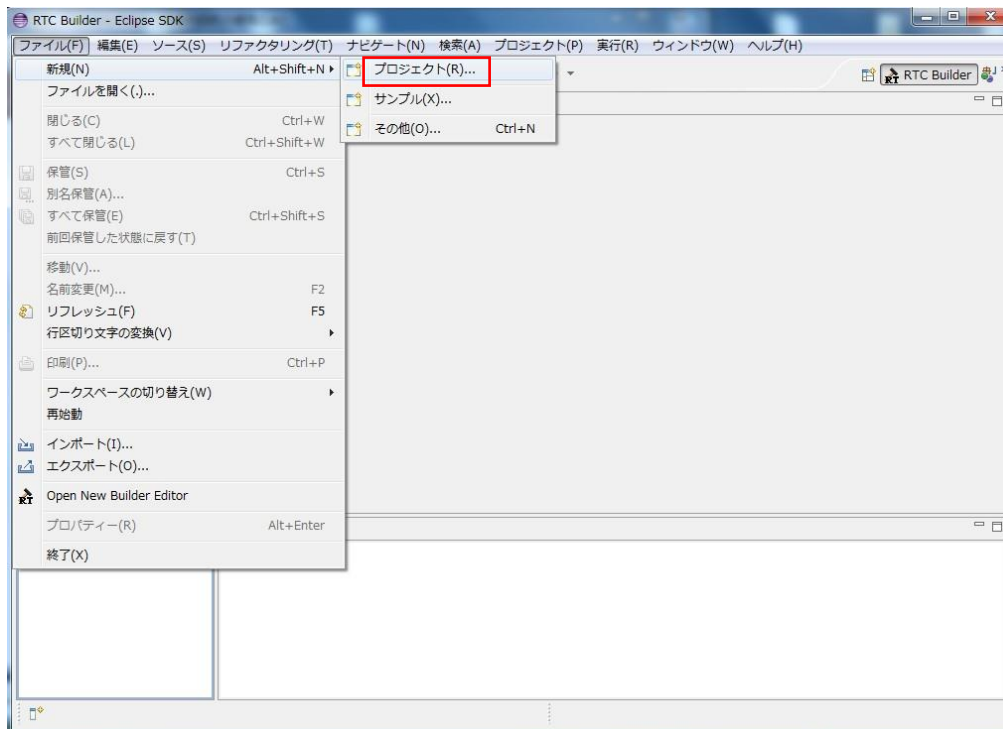


「RTC Builder」を選択することで、RTCBuilder が起動します。メニューバーに「カナヅチと RT」の RTCBuilder のアイコンが現れれば完了です。

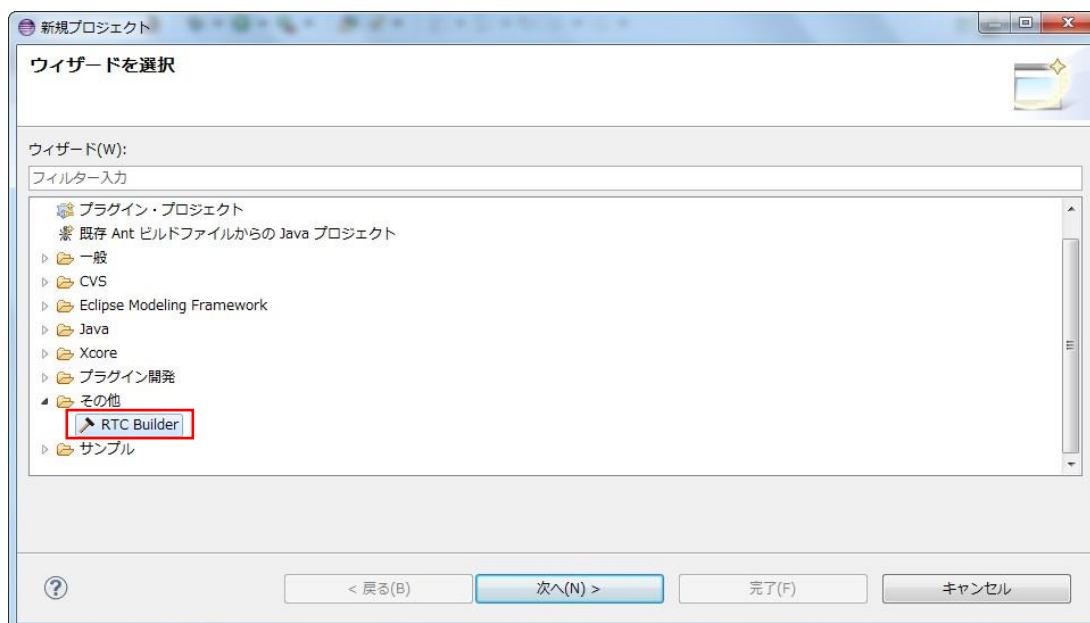
コンポーネント作成

3.2 新規プロジェクトを作成する

画面上部のメニューから[ファイル]―[新規]―[プロジェクト]を選択します。

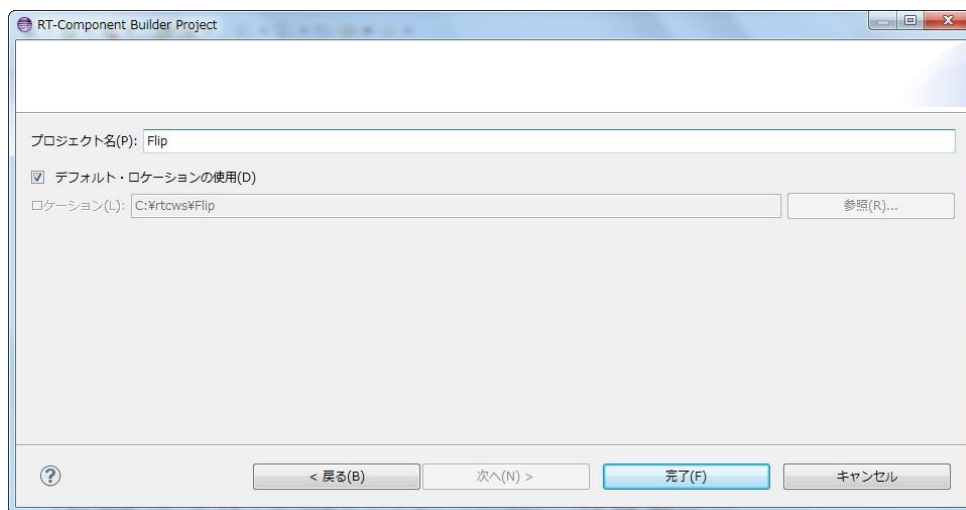


「新規プロジェクト」画面で「その他」―「RTC Builder」を選択し、「次へ」をクリックします。

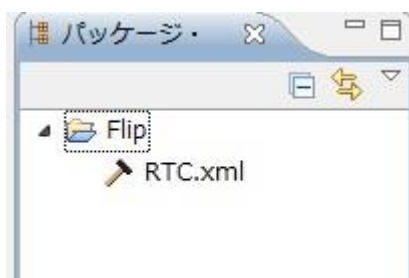


コンポーネント作成

「プロジェクト名」欄に作成するプロジェクト名（ここでは Flip）を入力して「完了」をクリックします。



下記画面の様にパッケージエクスプローラ内にプロジェクトが追加されれば完了です。



3.3 RTC プロファイルエディタを起動する

基本的に RTC.xml が生成された時点で、RTCBUILDER のエディタが開きます。

もし開かない場合は、「カナヅチと RT」の RTCBuilder のアイコンを押下するか、メニューバーの [ファイル]-[Open New Builder Editor] を選択します。

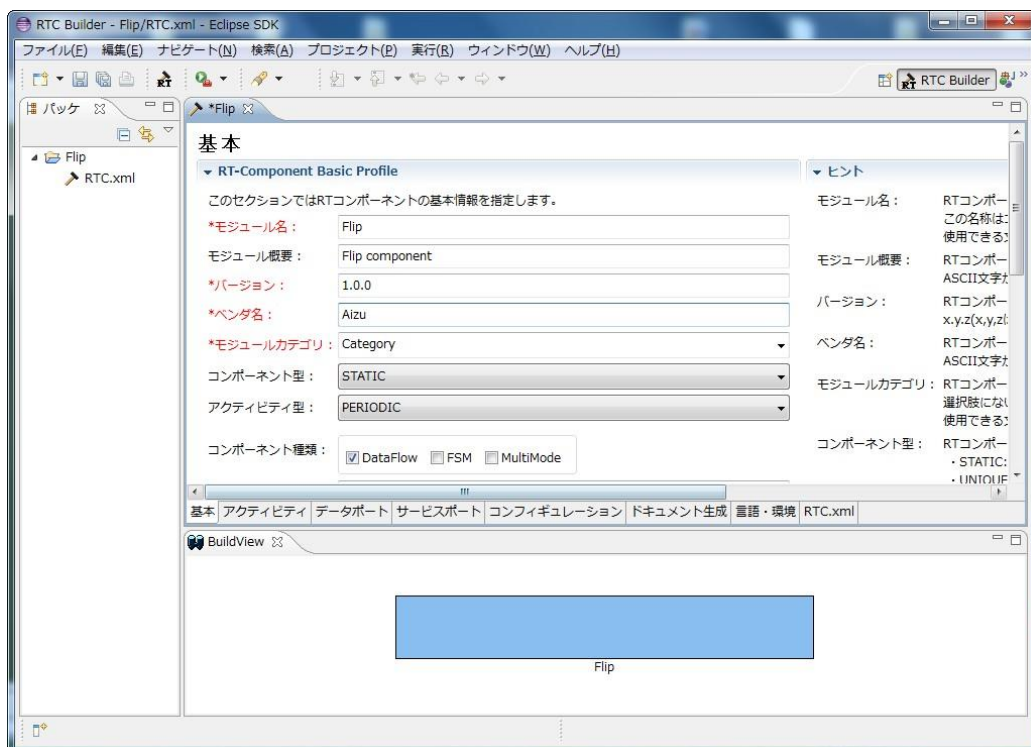


3.4 プロファイルの情報を入力しコードを生成する

3.4.1 基本

一番左の「基本」タブを選択し、基本情報を設定します。コンポーネントの名前や概要などを記入します。ラベルが赤文字の項目は必須項目です。その他はデフォルトのままで大丈夫です。

モジュール名:Flip
モジュール概要:Flip component
バージョン:1.0.0
ベンダ名:Aizu
モジュールカテゴリ:Category
コンポーネント型:STATIC
アクティビティ型:PERIODIC
コンポーネント種類:DataFlowComponent
最大インスタンス数:1
実行型:PeriodicExecutionContext
実行周期:1000.0

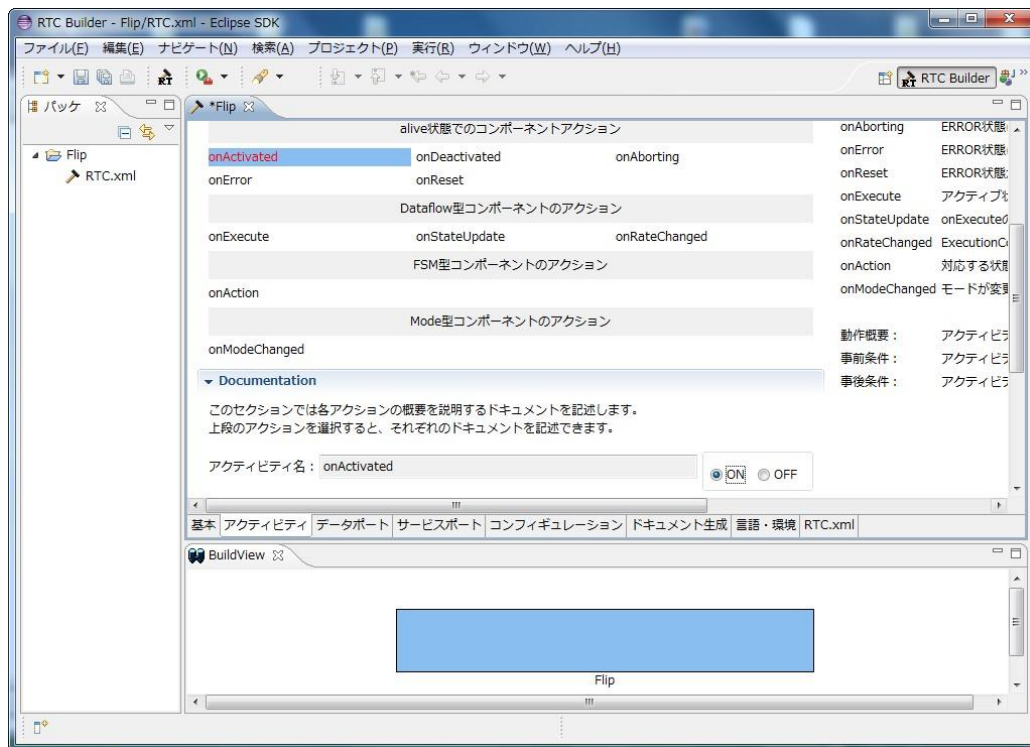


コンポーネント作成

3.4.2 アクティビティ

次に、「アクティビティ」タブを選択し、使用するアクションコールバックを指定します。

Flip コンポーネントでは、onActivated(),onDeactivated(),onExecute()コールバックを使用します。下図のように赤枠の onAtivated をクリック後に赤枠のラジオボタンにて"on"にチェックを入れます。onDeactivated,onExecute についても同様の手順を行います。



最終的に下図のようになります。



コンポーネント作成

3.4.3 データポート

「データポート」タブを選択し、データポートの情報を入力します。先ほど決めた仕様を元に以下のように入力します。[Add]ボタンを押して新しいデータポートを追加します。

・ InPort

ポート名: originalImage

データ型:

RTC::CameraImage

変数名: originalImage

表示位置: left

・ OutPort

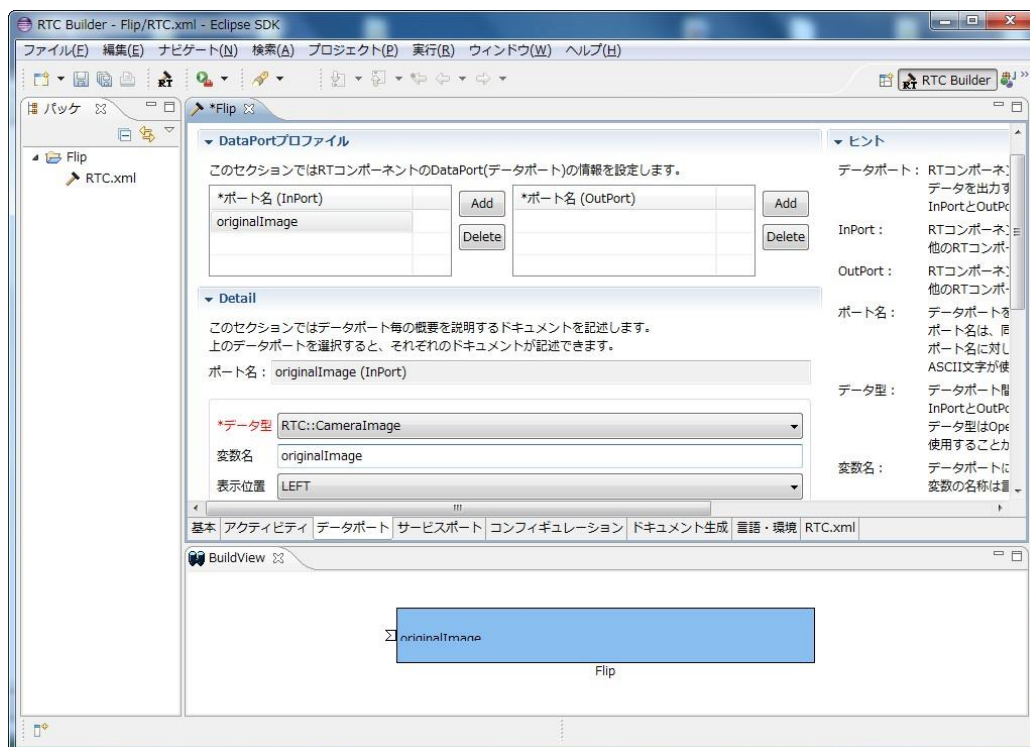
ポート名: flippedImage

データ型:

RTC::CameraImage

変数名: flippedImage

表示位置: right



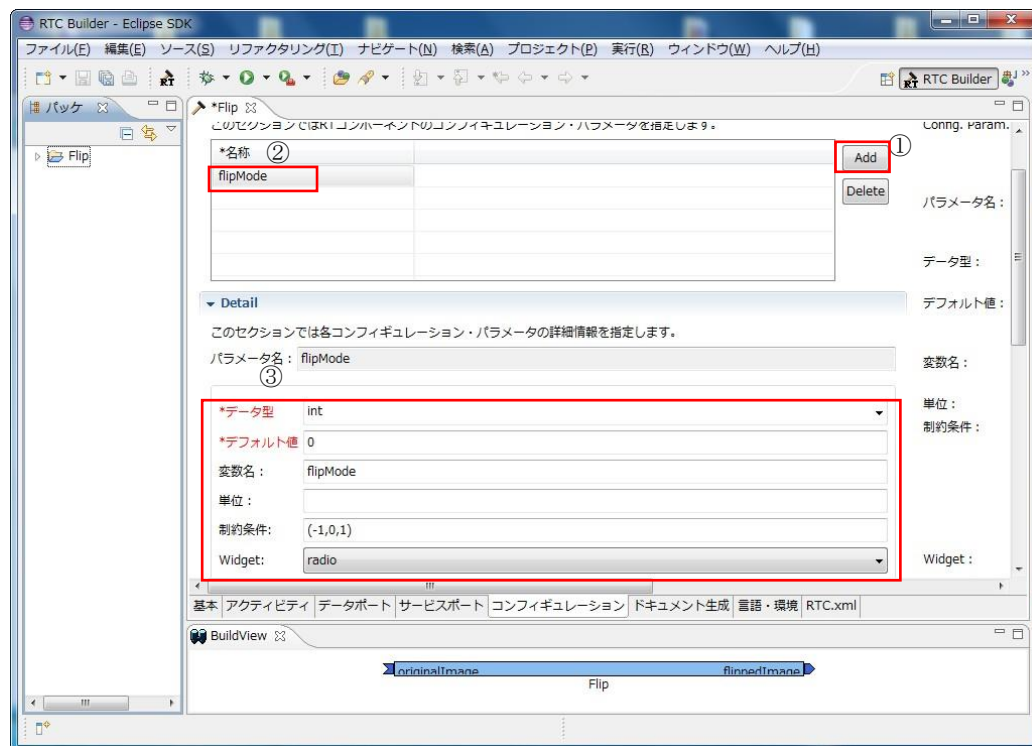
コンポーネント作成

3.4.4 コンフィギュレーション

「コンフィギュレーション」タブを選択し、先ほど決めた仕様を元に、**Configuration** の情報を入力します。制約条件および **Widget** とは、**RTSystemEditor** でコンポーネントのコンフィギュレーションパラメータを表示する際に **GUI** で値の変更を行うための形式を表すものです。

ここでは、**flipMode** の値は先ほど仕様を決めたときに、**-1,0,1** の 3 つの値のみ取ることにしたので、ラジオボタンを使用することになります。**[Add]** ボタンを押して新しいコンフィギュレーションを追加します。

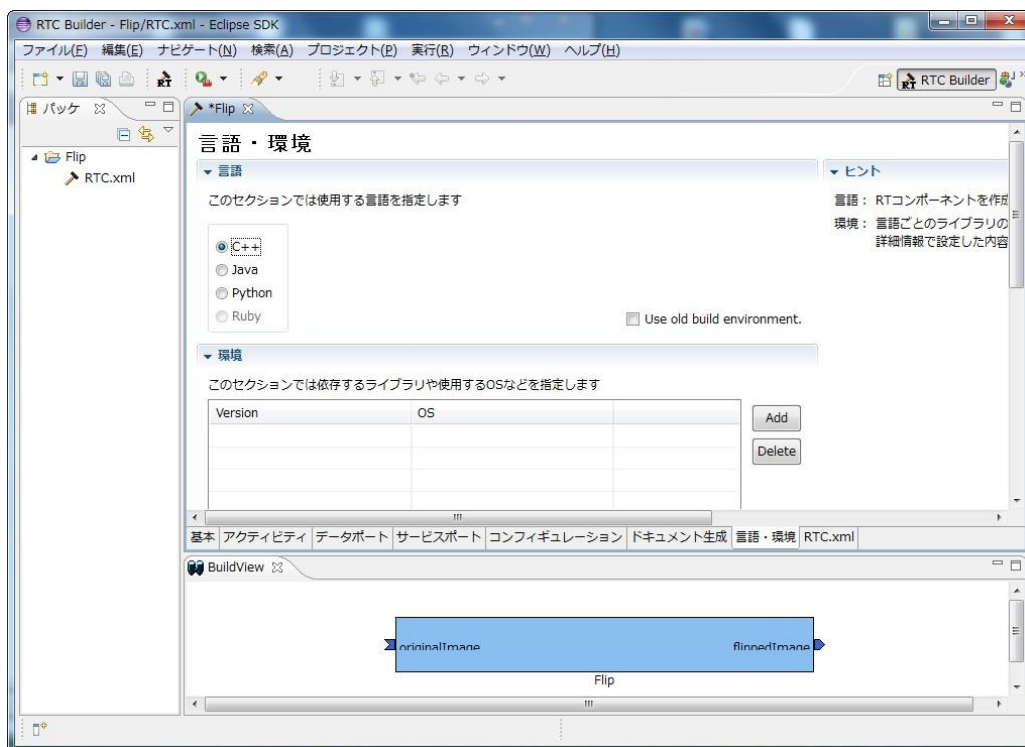
名称: **flipMode**
データ型: **int**
デフォルト値: **1**
変数名: **flipMode**
制約条件: **(-1, 0, 1)**
Widget: **radio**



コンポーネント作成

3.4.5 言語・環境

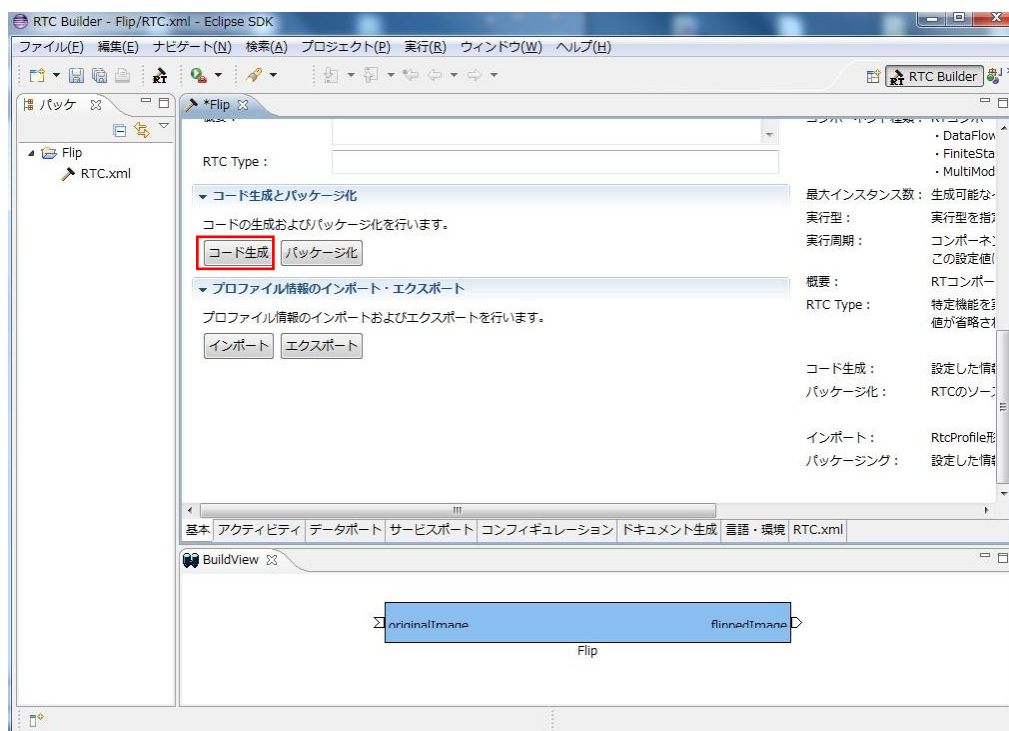
「言語・環境」タブを選択し、プログラミング言語を選択します。ここでは、C++(言語)を選択します。言語・環境はデフォルトでは設定されていないので、指定し忘れるとコード生成時にエラーになりますので、必ず言語の指定を行うようにしてください。



コンポーネント作成

3.4.6 コード生成

全ての設定が完了したら、「基本」タブに戻りコード生成ボタンをクリックします。問題がなければコンポーネントの雛型が生成されます。



コンポーネント作成

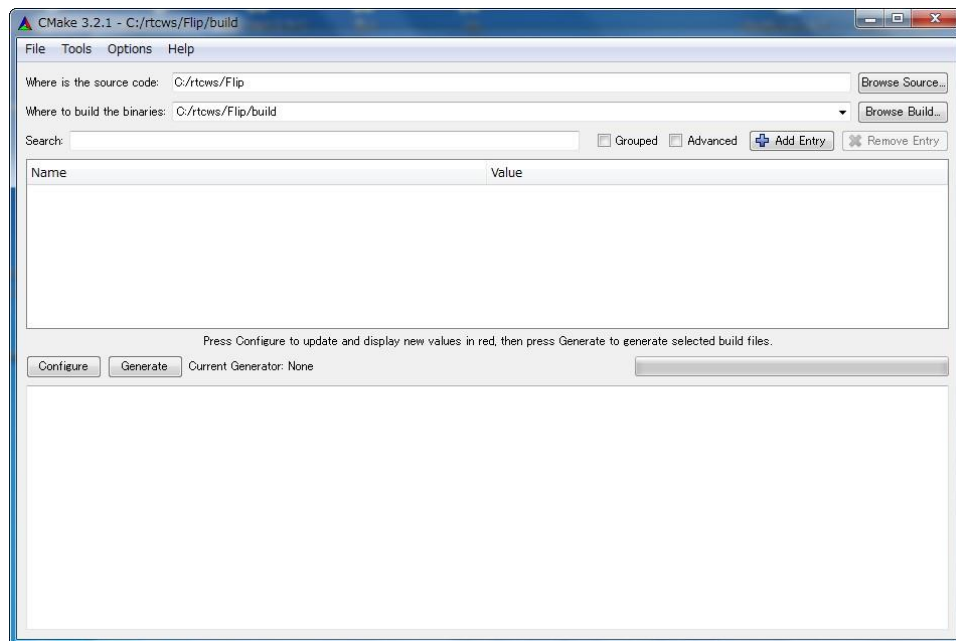
3.5 仮ビルドする

ここまでの作業で Flip コンポーネントの雛型が生成されました。

次の作業として CMake を利用してビルド環境の Configure を行います。

3.5.1 CMake を起動する

スタートメニューなどから CMake (cmake-gui) を起動します。



3.5.2 値を指定する

画面上部に以下のようなテキストボックスがあります。

- Where is the source code
- Where to build the binaries

「Where is the source code」に CMakeList.txt が有る場所、「Where to build the binaries」にビルドディレクトリを指定します。

CMakeList.txt はデフォルトでは<ワークス ペースディレクトリ>/Flip になります。

ビルドディレクトリとは、ビルドするためのプロジェクトファイルやオブジェクトファイル、バイナリを格納する場所のことです。場所は任意ですが、この場合 <ワークスペースディレクトリ>/Flip/build のように分かりやすい名前をつけた Flip のサブディレクトリを指定することをお勧めします。

ディレクトリは自動で作成されるので指定前に作成する必要はありません。

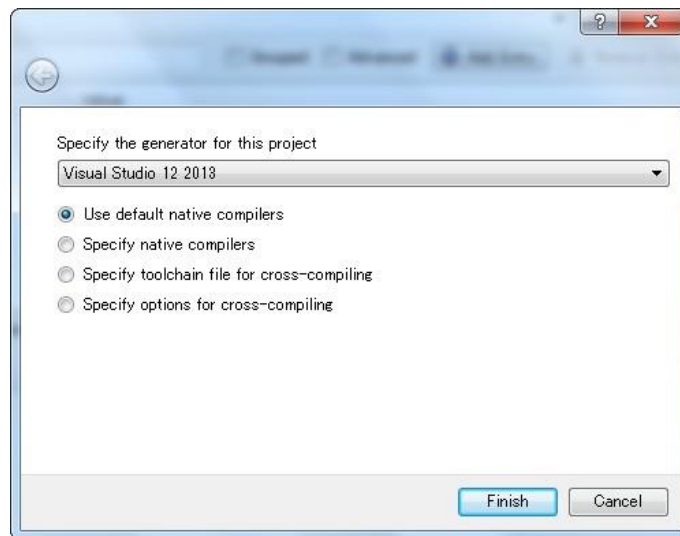
コンポーネント作成

今回は以下の様に設定します。

Where is the source code	C:\Yrtcws\F\Flip
Where to build the binaries	C:\Yrtcws\F\Flip\build

3.5.3 プロジェクト種類を指定する

指定したら、下の **Configure** ボタンを押します。すると下図のようなダイアログが表示されますので、生成したいプロジェクトの種類を指定します。



Visual Studio バージョン	32/64 bit	生成したいプロジェクトの種類
Visual Studio 2013	32 bit	Visual Studio 12 2013
	64 bit	Visual Studio 12 2013 Win 64
Visual Studio 2015	32 bit	Visual Studio 14 2015
	64 bit	Visual Studio 14 2015 Win 64

3.5.4 Configure と Generate

ダイアログで **Finish** を押すと **Configure** が始まります。問題がなければ下部のログウィンドウに **Configuring done** と出力されますので、続けて **Generate** ボタンを押します。
Generating done と出ればプロジェクトファイル・ソリューションファイル等の出力が完了します。

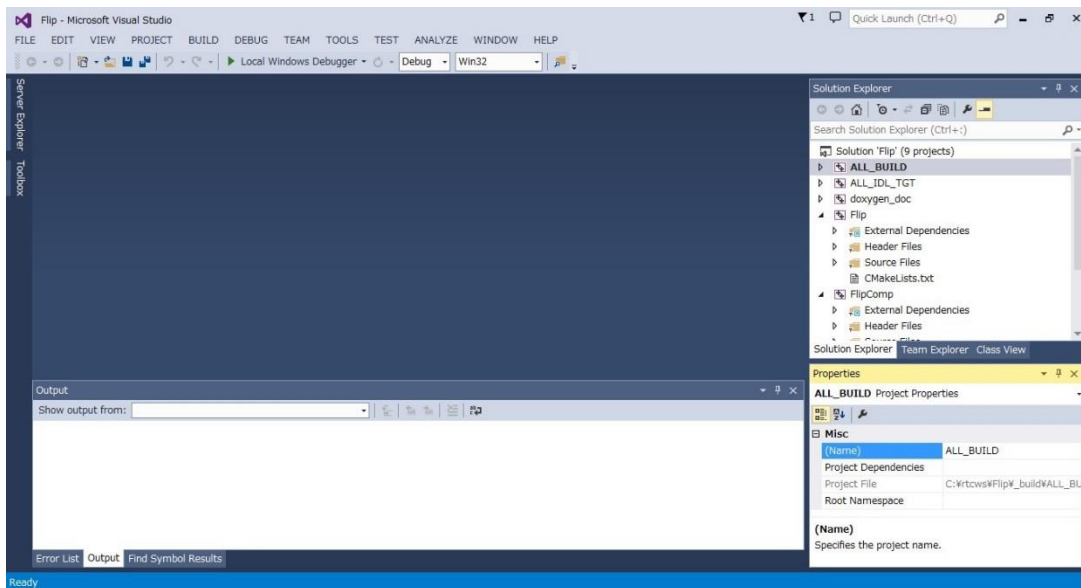
3.5.5 Visual Studio を起動する

次に先ほど指定した build ディレクトリの中の **Flip.sln** をダブルクリックして Visual Studio 2013 を起動します。

コンポーネント作成

3.5.6 ALL_BUILD を実行する

起動後、ソリューションエクスプローラーの ALL_BUILD を右クリックしビルドを選択してビルドします。特に問題がなければ正常にビルドが終了します。

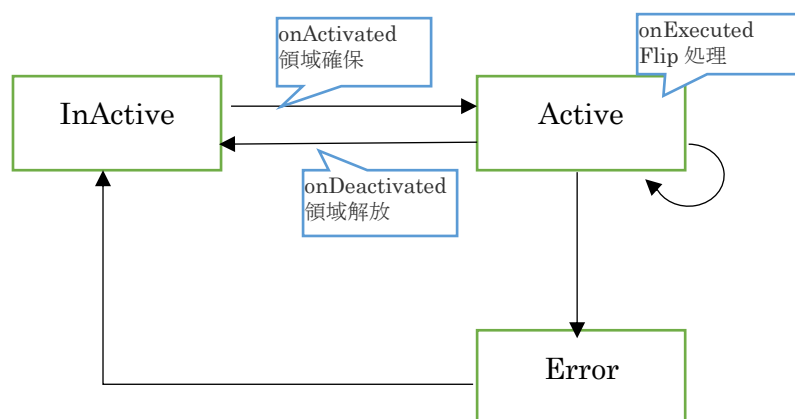


4 ヘッダ、ソースを編集する

4.1 アクティビティ処理を実装する

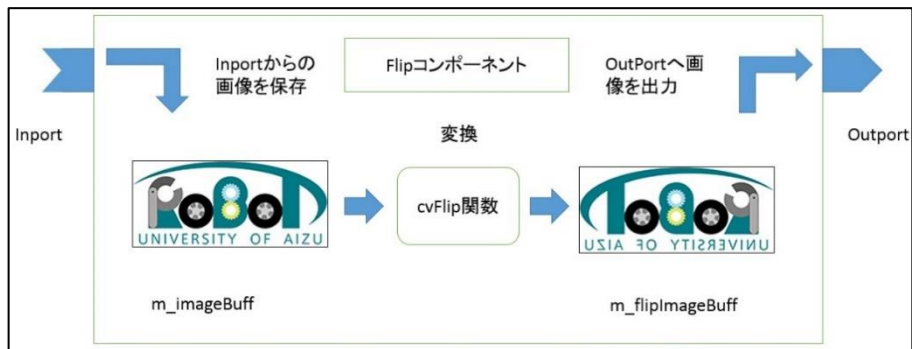
Flip コンポーネントでは、InPort から受け取った画像を画像保存用バッファに保存し、その保存した画像を OpenCV の `cvFlip()` 関数にて変換します。その後、変換された画像を OutPort から送信します。

`onActivated()`, `onExecute()`, `onDeactivated()` での処理内容は下図のようになります。



コンポーネント作成

Flip コンポーネントの処理の流れは下図のようになります。



4.2 ヘッダファイル (Flip.h)を編集する

OpenCV のライブラリを使用するため、OpenCV のライブラリファイルをインクルードします。下記内容をインクルードしている所に追加してください。

```
//OpenCV 用ライブラリファイルのインクルード
#include<cv.h>
#include<cxcore.h>
#include<highgui.h>
```

画像の保存用にメンバー変数を追加します。下記内容を class の private:の中(// <rtc-template block="private_attribute">の下)に追加してください。

```
IplImage* m_imageBuff;
IplImage* m_flipImageBuff;
```


4.3 ソースファイル (Flip.cpp)を編集する

下記のように、onActivated(),onDeactivated(),onExecute()を実装します。

4.3.1 onActivated()

```
RTC::ReturnCode_t Flip::onActivated(RTC::UniqueId ec_id)
{
    // イメージ用メモリの初期化
    m_imageBuff = NULL;
    m_flipImageBuff = NULL;

    // OutPort の画面サイズの初期化
    m_flippedImage.width = 0;
    m_flippedImage.height = 0;

    return RTC::RTC_OK;
}
```

4.3.2 onDeactivated()

```
RTC::ReturnCode_t Flip::onDeactivated(RTC::UniqueId ec_id)
{
    if(m_imageBuff != NULL)
    {
        // イメージ用メモリの解放
        cvReleaseImage(&m_imageBuff);
        cvReleaseImage(&m_flipImageBuff);
    }

    return RTC::RTC_OK;
}
```

コンポーネント作成

4.3.3 onExecute()

```
RTC::ReturnCode_t Flip::onExecute(RTC::UniqueId ec_id)
{
    // 新しいデータのチェック
    if (m_originalImageIn.isNew()) {
        // InPort データの読み込み
        m_originalImageIn.read();

        // InPort と OutPort の画面サイズ処理およびイメージ用メモリの確保
        if( m_originalImage.width != m_flippedImage.width || m_originalImage.height !=
m_flippedImage.height)
        {
            m_flippedImage.width = m_originalImage.width;
            m_flippedImage.height = m_originalImage.height;
            // InPort のイメージサイズが変更された場合
            if(m_imageBuff != NULL)
            {
                cvReleaseImage(&m_imageBuff);
                cvReleaseImage(&m_flipImageBuff);
            }
            // イメージ用メモリの確保
            m_imageBuff = cvCreateImage(cvSize(m_originalImage.width, m_originalImage.height),
IPL_DEPTH_8U, 3);
            m_flipImageBuff = cvCreateImage(cvSize(m_originalImage.width, m_originalImage.height),
IPL_DEPTH_8U, 3);
        }

        // InPort の画像データを IplImage の imageData にコピー
        memcpy(m_imageBuff->imageData, (void
*&(m_originalImage.pixels[0]), m_originalImage.pixels.length()));

        // InPort からの画像データを反転する。 m_flipMode 0: X 軸周り, 1: Y 軸周り, -1: 両方の軸周り
        cvFlip(m_imageBuff, m_flipImageBuff, m_flipMode);

        // 画像データのサイズ取得
        int len = m_flipImageBuff->nChannels * m_flipImageBuff->width * m_flipImageBuff->height;
        m_flippedImage.pixels.length(len);

        // 反転した画像データを OutPort にコピー
        memcpy((void *)&(m_flippedImage.pixels[0]), m_flipImageBuff->imageData, len);

        // 反転した画像データを OutPort から出力する。
        m_flippedImageOut.write();
    }
    return RTC::RTC_OK;
}
```

4.4 CMake でビルドに必要なファイルを生成する

C:\rtcws\Fip\src\CMakeLists.txt を編集します。

このコンポーネントでは OpenCV を利用していますので、OpenCV のヘッダのインクルードパス、ライブラリやライブラリサーチパスを与える必要が有ります。以下の 2 点を追加・変更するだけで OpenCV のライブラリがリンクされ使えるようになります。

- ・ find_package(OpenCV REQUIRED)を追加
- ・ 最初の target_link_libraries に \${OpenCV_LIBS} を追加

※target_link_libraries は 2 ヶ所あります。

※追加するときは\${OpenCV_LIBS}の前に半角スペースを入れてください。

```
set(comp_srcs Flip.cpp )
set(standalone_srcs FlipComp.cpp)

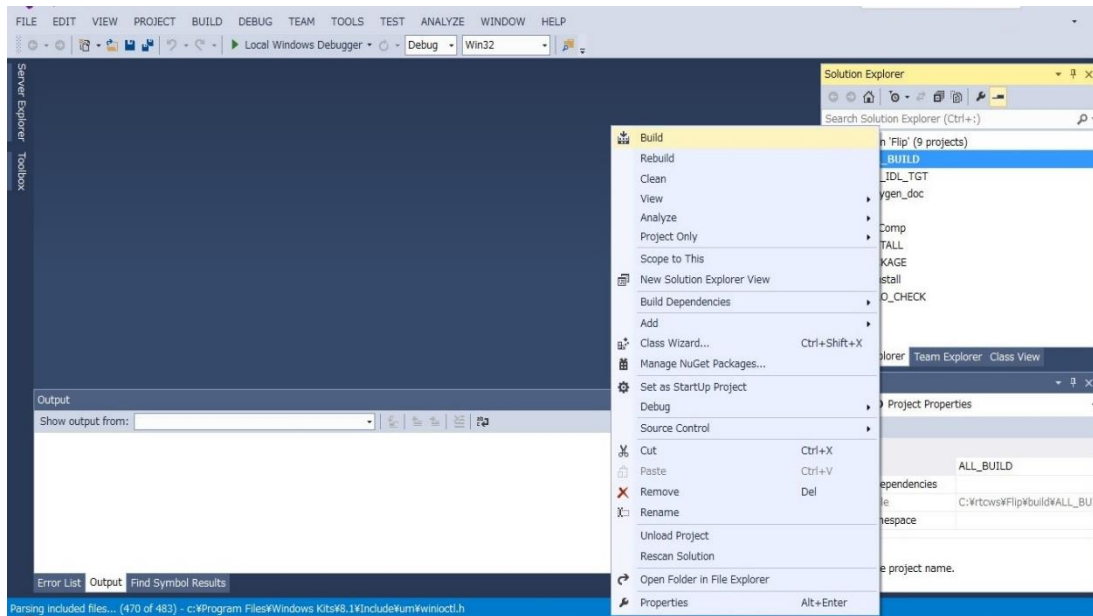
find_package(OpenCV REQUIRED) ←この行を追加
: 中略
add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES} ${OpenCV_LIBS}) ←OpenCV_LIBS を追加
: 中略
add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
    ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES} ${OpenCV_LIBS}) ←OpenCV_LIBS を追加
```

コンポーネント作成

4.5 VC++でビルドする

CMakeList.txt を編集したので、再度 CMake GUI で Configure および Generate を行います。CMake の Generate が正常に終了した事を確認し、Flip.sln ファイルをダブルクリックし、Visual C++ 2013 を起動します。

Visual C++ 2013 の起動後、下図のように右クリックでコンポーネントのビルドを行います。



4.6 コンポーネントを起動する

4.6.1 NameService を起動する

コンポーネントの参照を登録するためのネームサービスを起動します。

[スタート]メニューから[すべてのプログラム]→[OpenRTM-aist x.y]→ [tools]→[Start Naming Service]をクリックしてください。

※ [Start Naming Service]をクリックしても omniNames が起動されない場合は、フルコンピュータ名が 14 文字以内に設定されているかを確認してください。

※ OpenRTM-aist C++ 1.1.1 使用の方は[Start C++ Naming Service]をクリックしてください。

※ Windows8 の場合下記パスを参考にあります。

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\OpenRTM-aist x.y\Tools

コンポーネント作成

4.6.2 Flip コンポーネントを起動する

C:\rtcws\Flip\build\src\Debug の FlipComp.exe をダブルクリックして起動します。

4.6.3 カメラコンポーネントとビューアコンポーネントを起動する

USB カメラのキャプチャ画像を OutPort から出力する OpenCVCameraComp と InPort で受け取った画像を画面に表示する CameraViewerComp を起動します。

[スタート]メニューから[すべてのプログラム]→[OpenRTM-aist x.y]→ [C++]→[Components] →[OpenCV-Examples] 内にあるのでダブルクリックで起動してください。

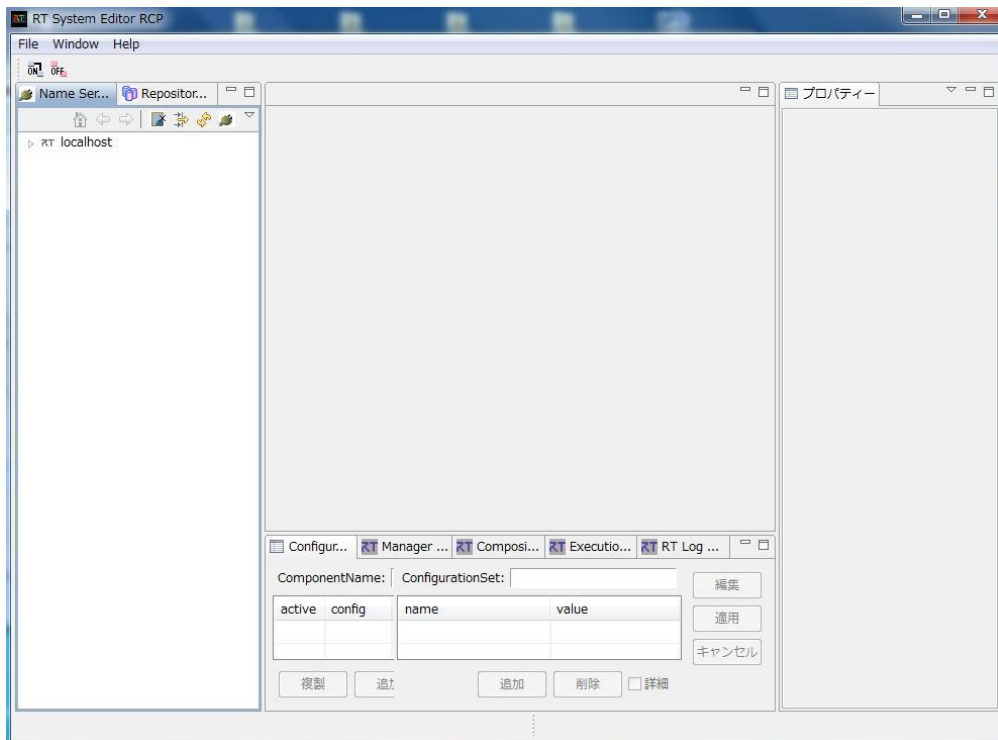
4.6.4 コンポーネントを接続する

WEB カメラから画像が取得できることを確認した後、Flip コンポーネントを確認します。

4.6.4.1 RTSystemEditor を起動する

最初に RTSystemEditor を起動します。

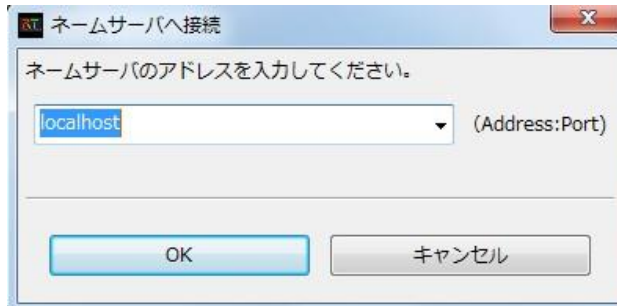
起動方法は RTCBuilder 画面右上の「パースペクティブを開く」を選択し、さらに[その他]を選択します。そして「パースペクティブ」の中から[RT System editor]を選択して起動させます。またはスタートメニューの「OpenRTM-aist x.y」→「tools」→[RTSystemEditorRCP]から起動します。



コンポーネント作成

4.6.4.2 コンポーネントを接続する

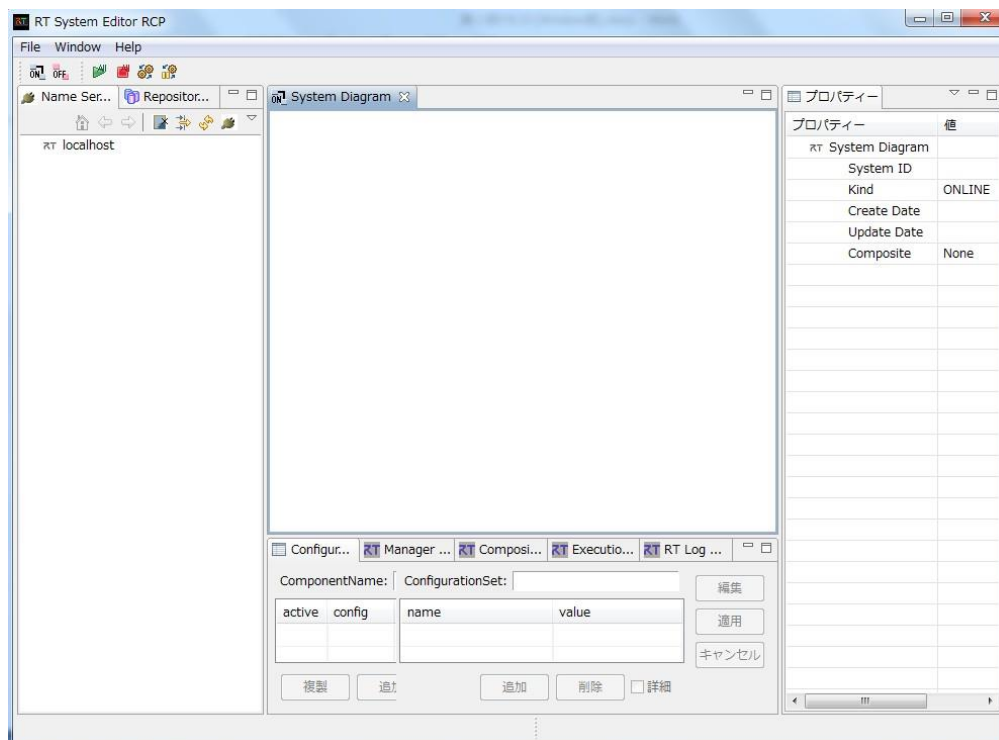
Name Service View に何も表示されていない場合は、RTSystemEditor の左側の Name Service View のコンセントアイコンをクリックし、ネームサーバへ接続します。表示された接続ダイアログに localhost と入力します。



Name Service View に[localhost]のリストが表示されます。

4.6.4.3 SystemDiagram

メニューバーの「Open New System Editor」 ボタンをクリックし、SystemDiagram を開きます。



コンポーネント作成

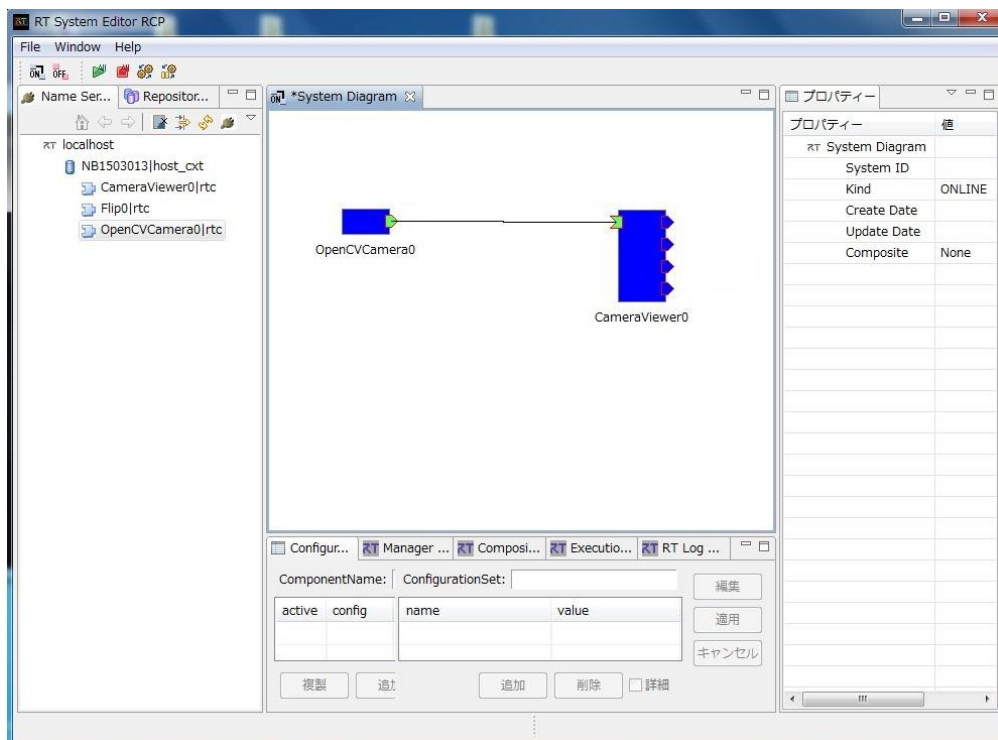
4.6.4.4 コンポーネントを配置する

次に Name Service View から OpenCVCameraComp と CameraViewerComp をドラックアンドドロップして SystemDiagram 上に配置してください。

次に、コンポーネントのデータポート同士を接続します。片方のデータポート上でドラッグすると線が伸びるので、接続したいデータポート上まで線を伸ばし接続します。

接続すると接続プロファイルが表示されるので OK をクリックします。

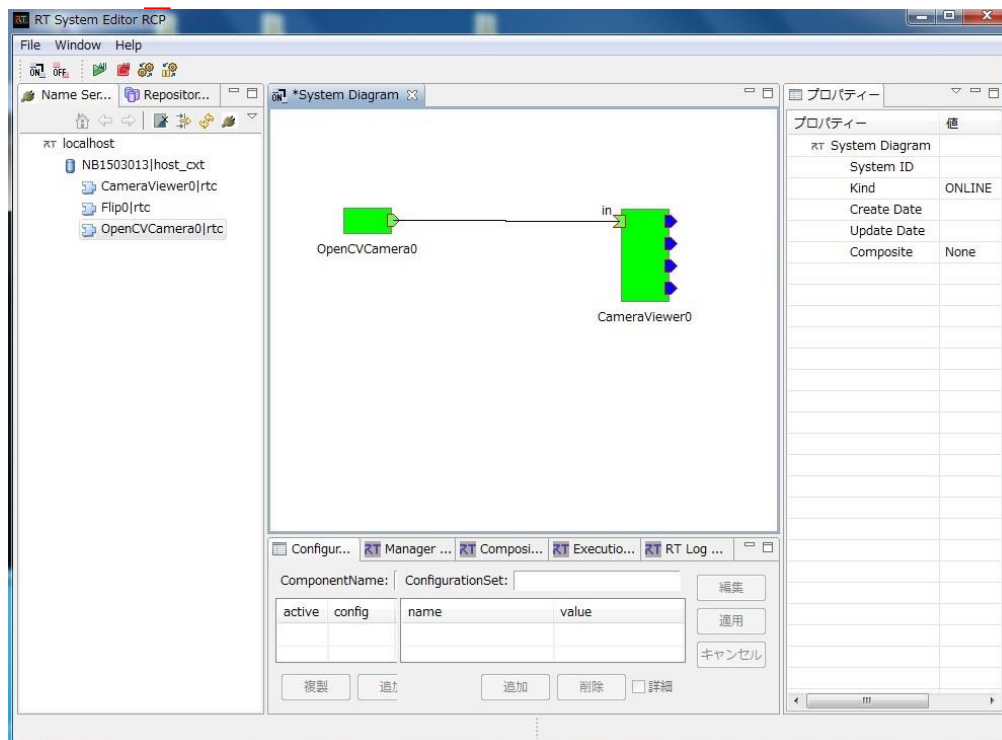
接続が完了すると下図の様になります。



コンポーネント作成

4.6.5 コンポーネントをアクティブ状態にする

RTSystemEditor の緑色の再生ボタンをクリックし、全てのコンポーネントをアクティブにします。正常にアクティブになると、下図のように黄緑色でコンポーネントが表示されます。

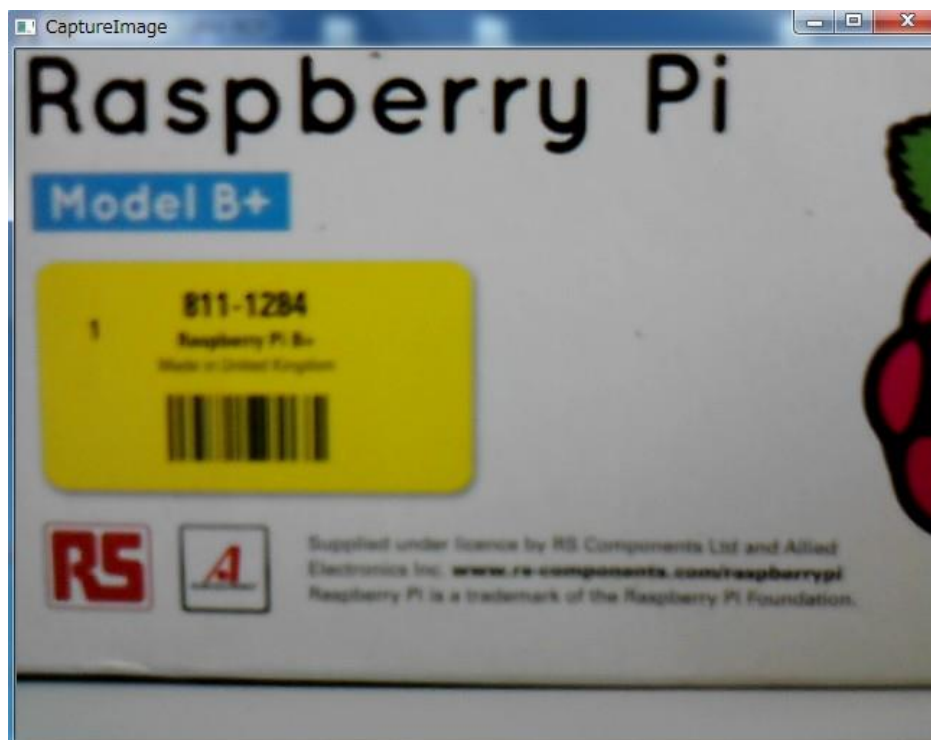


コンポーネント作成

4.6.6 動作確認

ウィンドウが出てきてカメラから取得した画像が表示されることを確認してください。

下の画像はカメラから取得したものです。色々なものを写し、画像が表示されることを確認してみてください。

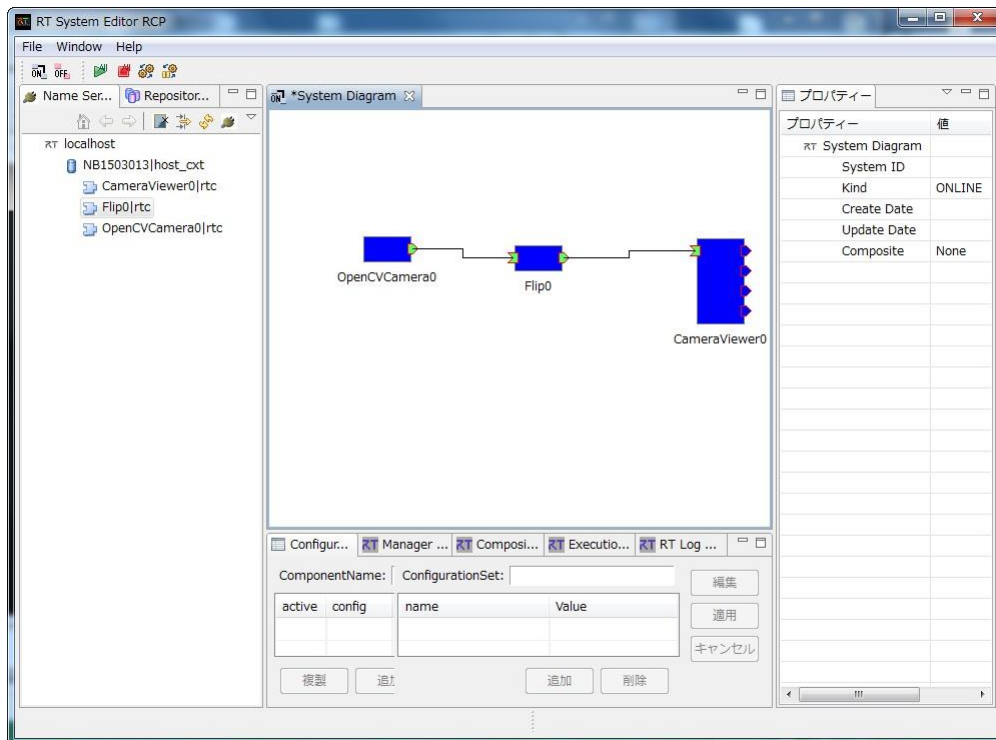


確認できましたら赤色の停止ボタンをクリックしてください。コンポーネントがディアクティブ状態になり、動作が終了します。

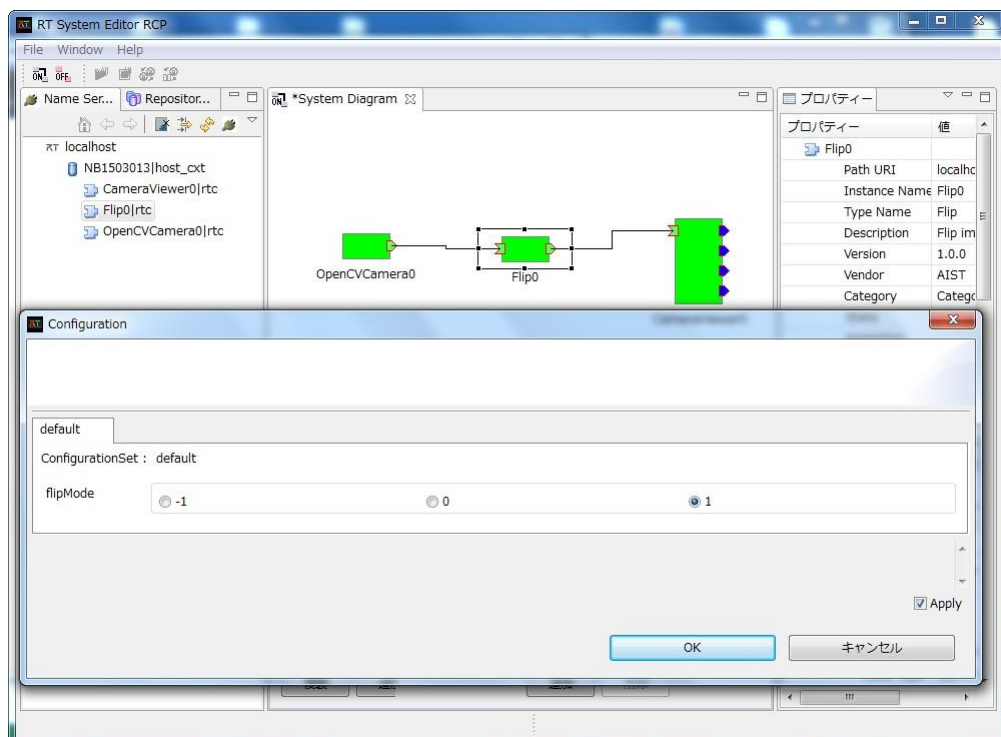
コンポーネント作成

4.6.7 Flip コンポーネントの動作確認

線を delete で消した後、下図の様に Flip コンポーネントを挿入して緑の再生ボタンをクリックしてください。



下図のようにコンフィギュレーションビューにて実行時にコンフィギュレーションを変更することができます。Flip コンポーネントをクリックしてコンフィギュレーションビューの編集を押すと下記ダイアログが表示されます。「flipMode」を「0」や「-1」などに変更し画像が反転することを確認してください。



5 Flip コンポーネントの全ソース

5.1 Flip コンポーネントソースファイル (Flip.cpp)

Flip.cpp のソースコードを以下に記載します。

Flip.cpp : http://192.168.11.101/wp/wp-content/uploads/2016/02/Flip_cpp.txt

5.2 Flip コンポーネントのヘッダファイル (Flip.h)

Flip.h のソースコードを以下に記載します。

Flip.h : http://192.168.11.101/wp/wp-content/uploads/2016/02/Flip_h.txt

5.3 Flip コンポーネントの全ソースコード

Flip コンポーネントの全ソースコードを以下に添付します。

Flip.zip : <http://192.168.11.101/wp/wp-content/uploads/2016/02/Flip.zip>