講習会テキスト第２部 Linux 版

目次

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 Linux 編）

http://www.openrtm.org/openrtm/ja/node/430 （2016/1/8 アクセス）

1. はじめに

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。

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。

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出来ます。

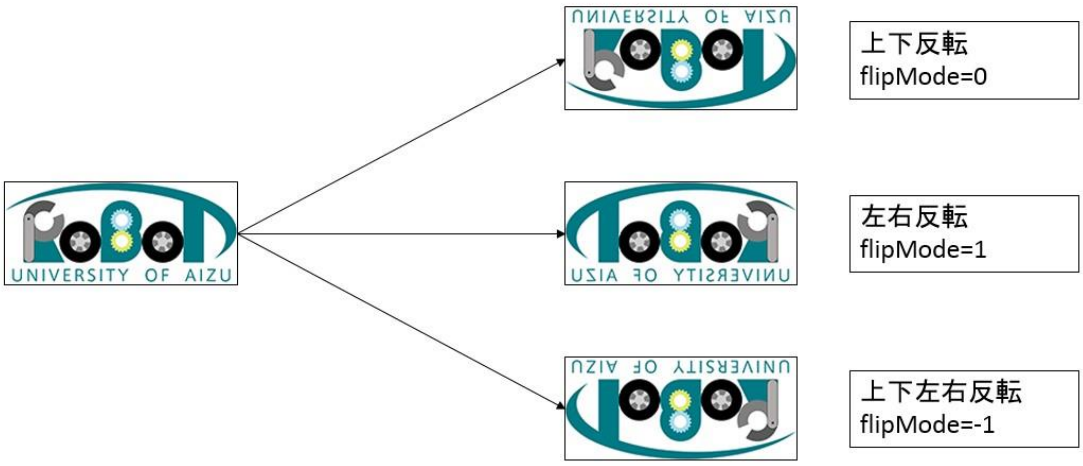
2. コンポーネントの仕様

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。

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 型を使用しています。

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 通りが有ります。これを実行時に指定できるように、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て指定できるようにします。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

それぞれに 0, 1, -1 を割り当てることにします。



以上から Flip コンポーネントの仕様をまとめると下記の様になります。

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の生成

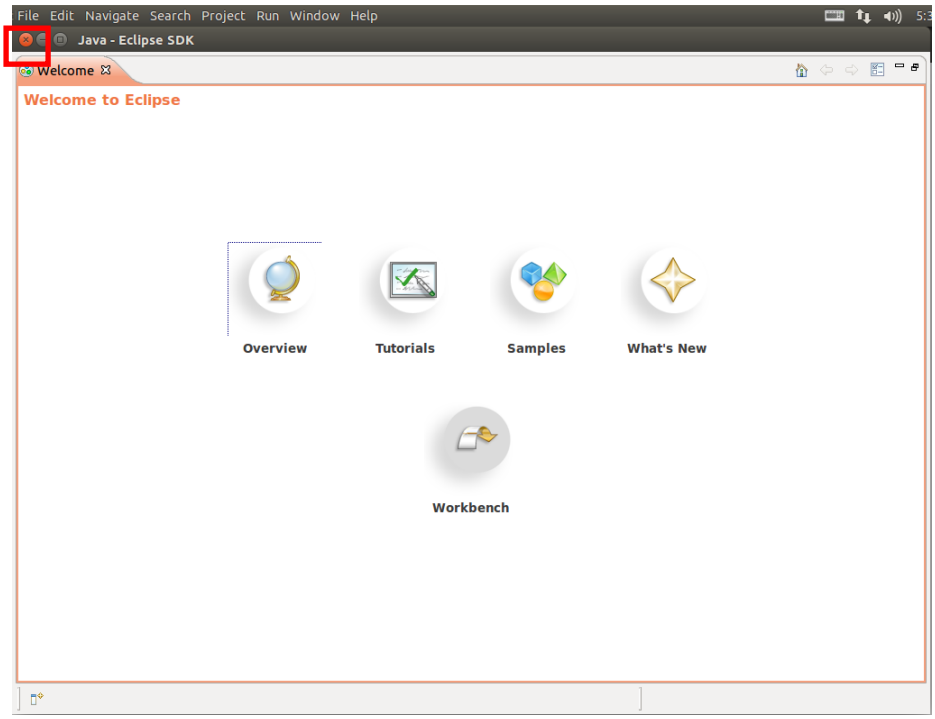
Flip コンポーネントの雛型の生成方法を説明します。

1. RTCBuilder の起動

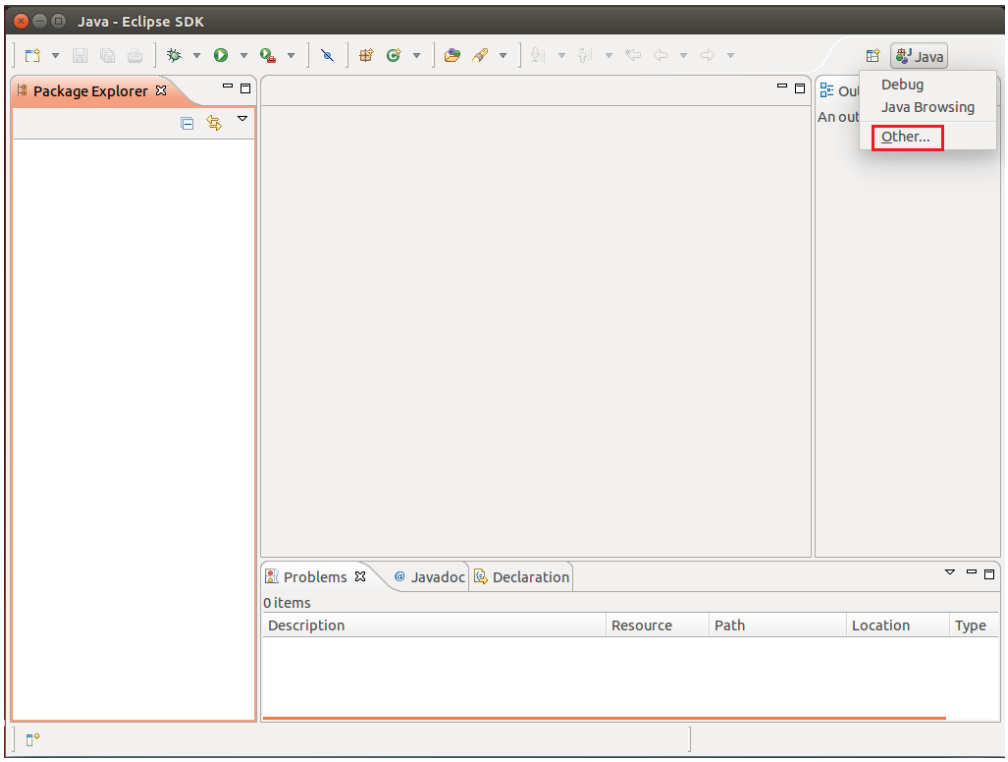
   OpenRTP を起動させると作成物を保存するディレクトリを指定します。ここでは下記ディレクトリに保存します。

[/home/<ユーザ名>/rtcws]

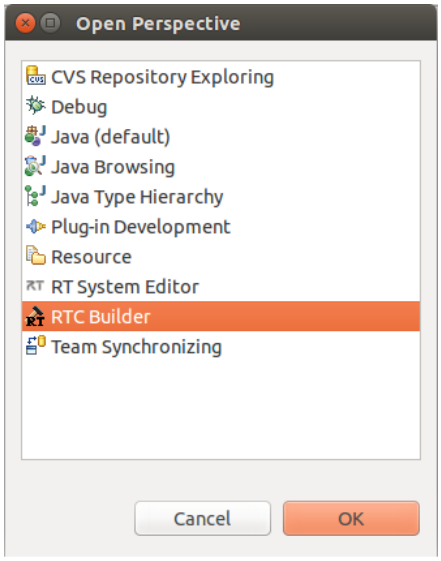
最初に起動したとき下記画面がでます。この画面は使用しないので左上の×ボタンを押します。



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



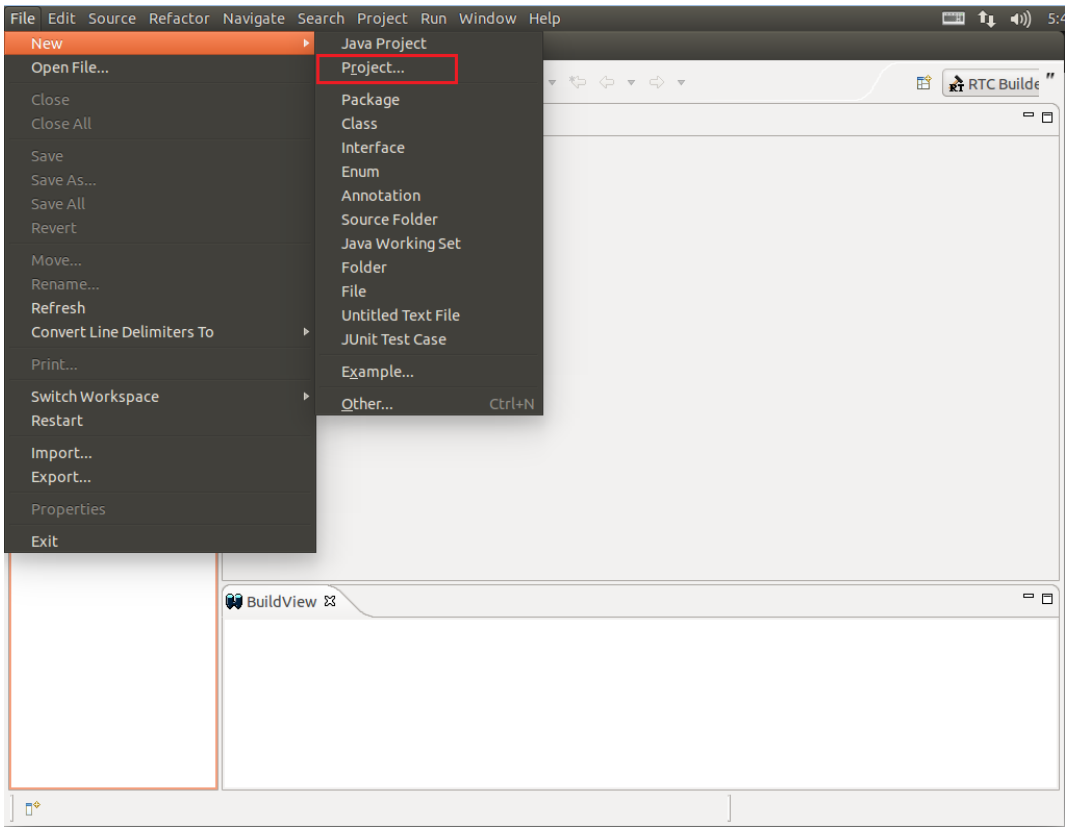
下記ウィンドウが出ますので[RTC Builder]を選択します。



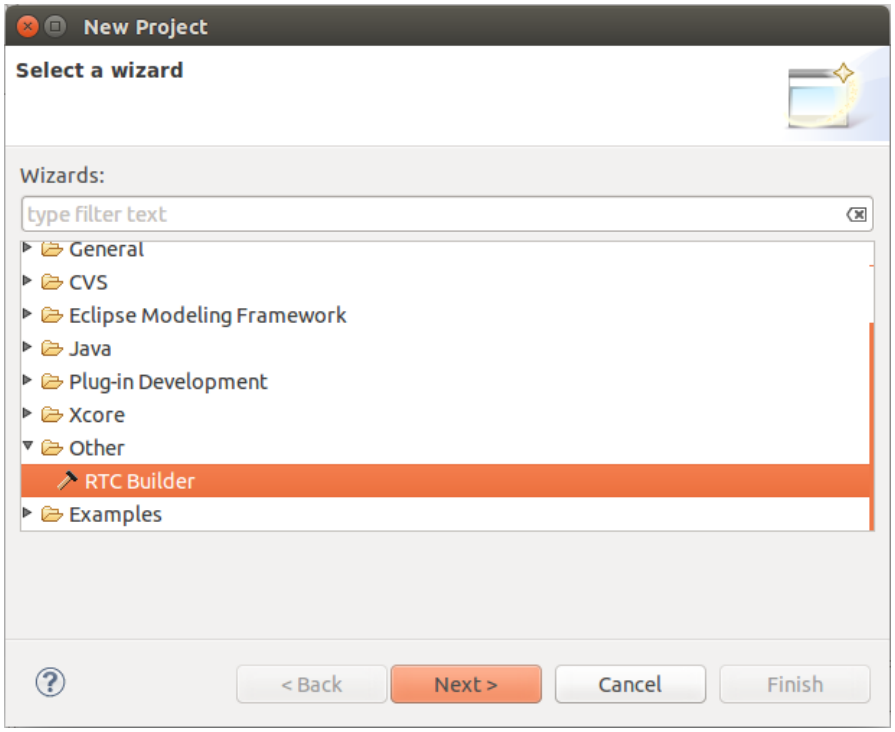
「RTC Builder」を選択することで、RTCBuilder が起動します。メニューバーに「カナヅチと RT」の RTCBuilder のアイコンが現れれば完了です。

2. 新規プロジェクトの作成

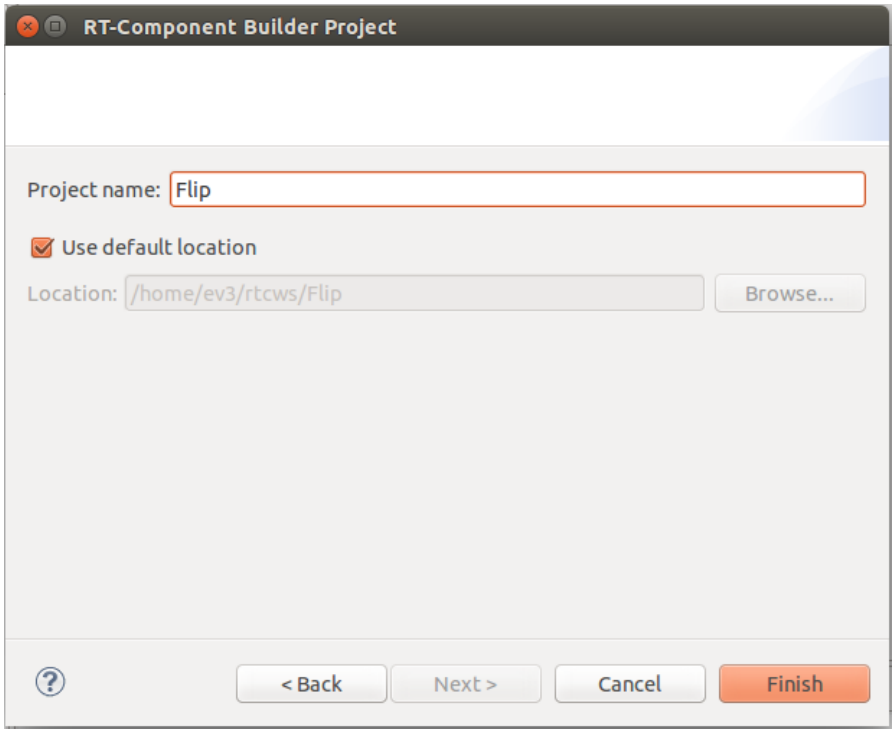
画面上部のメニューから[File]－[New]－[Project…]を選択



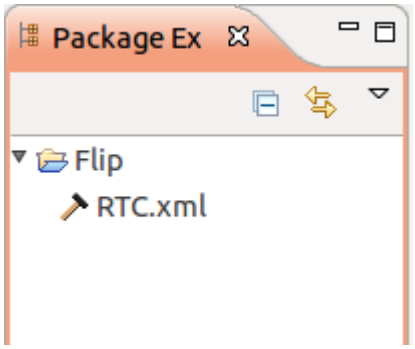
「New Project」画面において，「Other」－「RTC Builder」を選択し、「Next >」をクリック



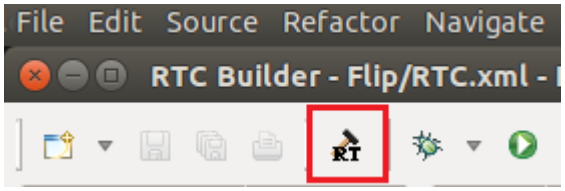
「Project name」欄に作成するプロジェクト名（ここでは Flip）を入力して「Finish」をクリックします。



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。



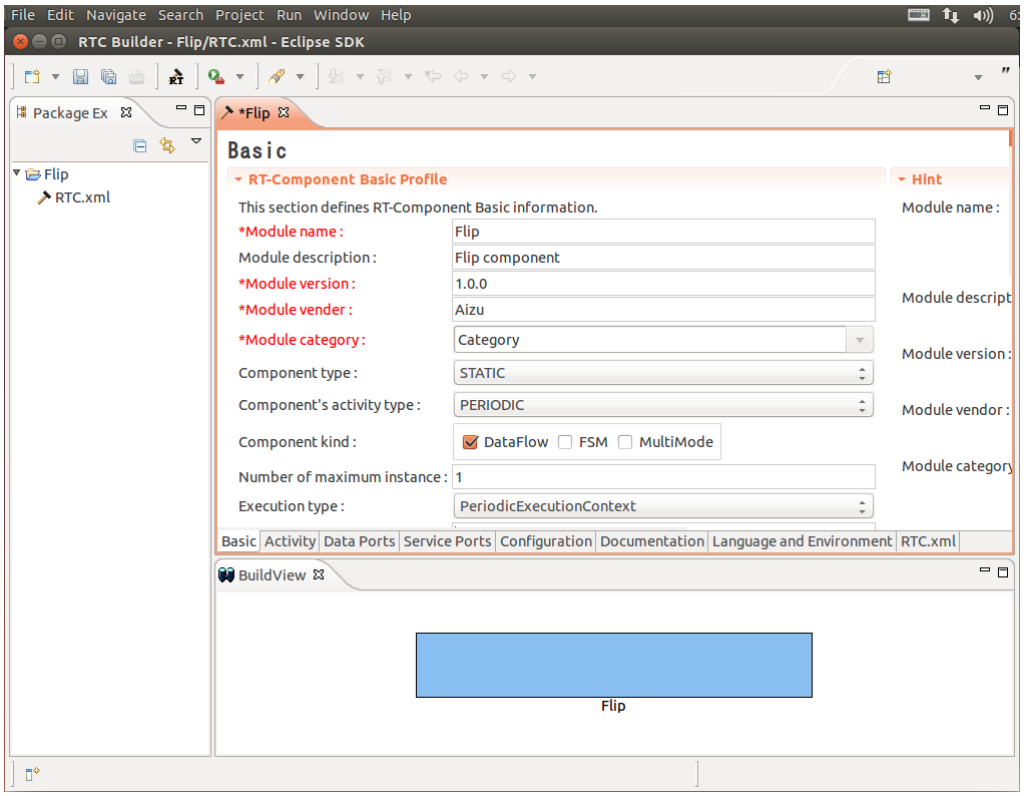
3. RTC プロファイルエディタの起動
   基本的に RTC.xml が生成された時点で、このプロジェクトに関連付けられているワークスペースとして RTCBuilder のエディタが開くはずです。
   もし開かない場合は、「カナヅチと RT」の RTCBuilder のアイコンを押下します。



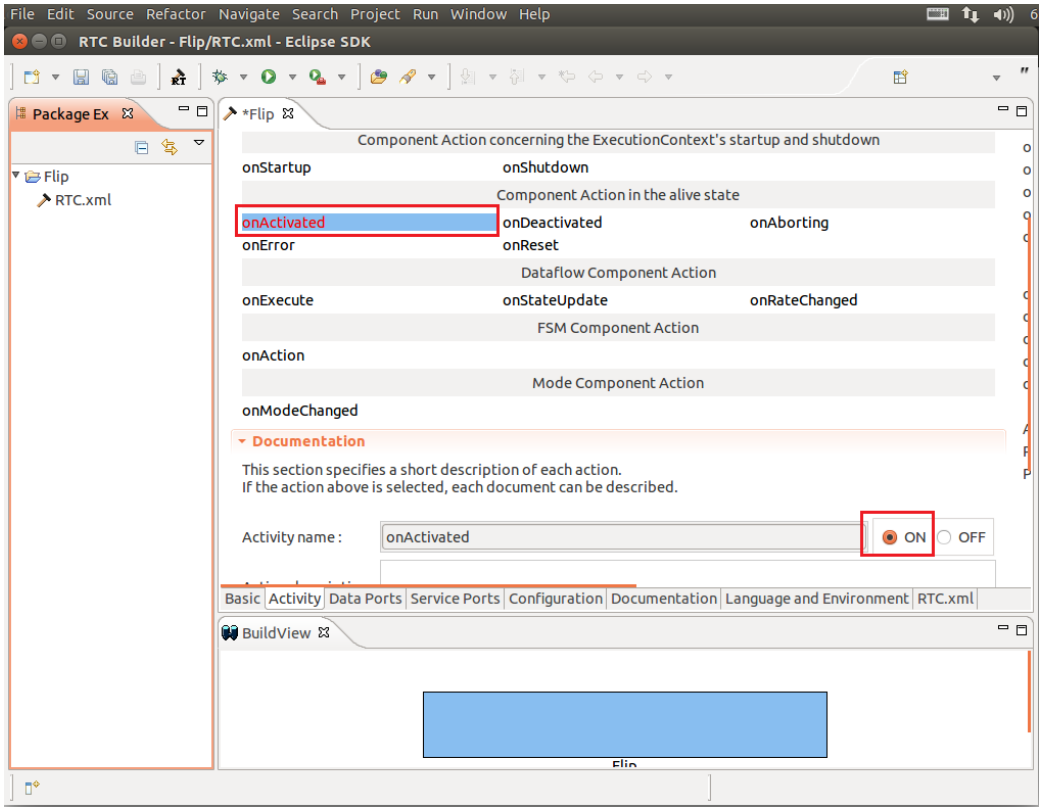
4. プロファイル情報入力とコードの生成

一番左の「Basic」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



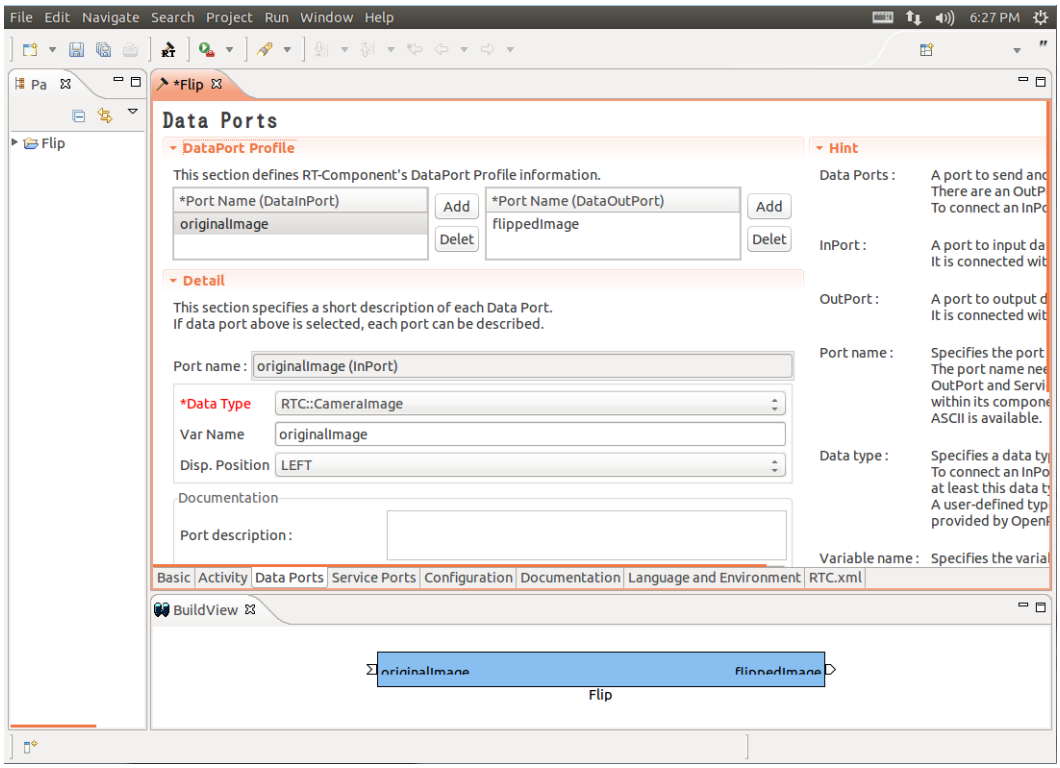
次に、「Activity」タブを選択し、使用するアクションコールバックを指定します。Flip コンポーネントでは、onActivated(),onDeactivated(),onExecute()コールバックを使用します。下図のように赤枠の onAtivated をクリック後に赤枠のラジオボタンにて"ON"にチェックを入れます。onDeactivated,onExecute についても同様の手順を行います。



「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。

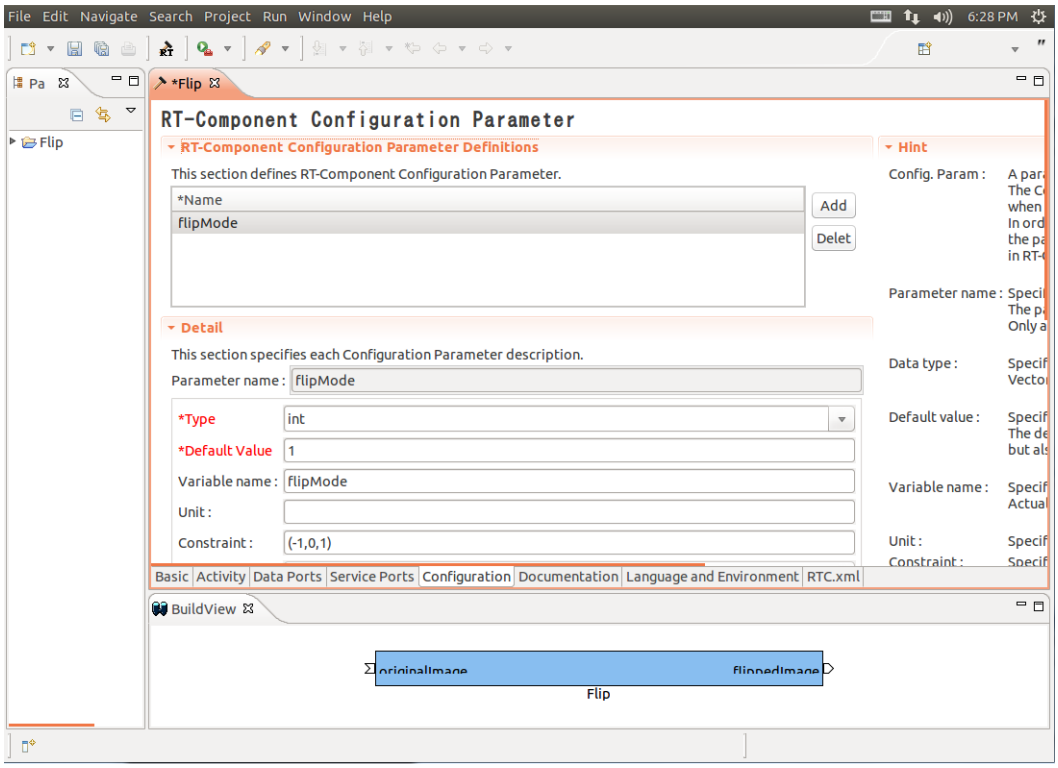
・InPort
ポート名: originalImage
データ型: RTC::CameraImage
変数名: originalImage
表示位置: LEFT

・OutPort
ポート名: flippedImage
データ型: RTC::CameraImage
変数名: flippedImage
表示位置: RIGHT

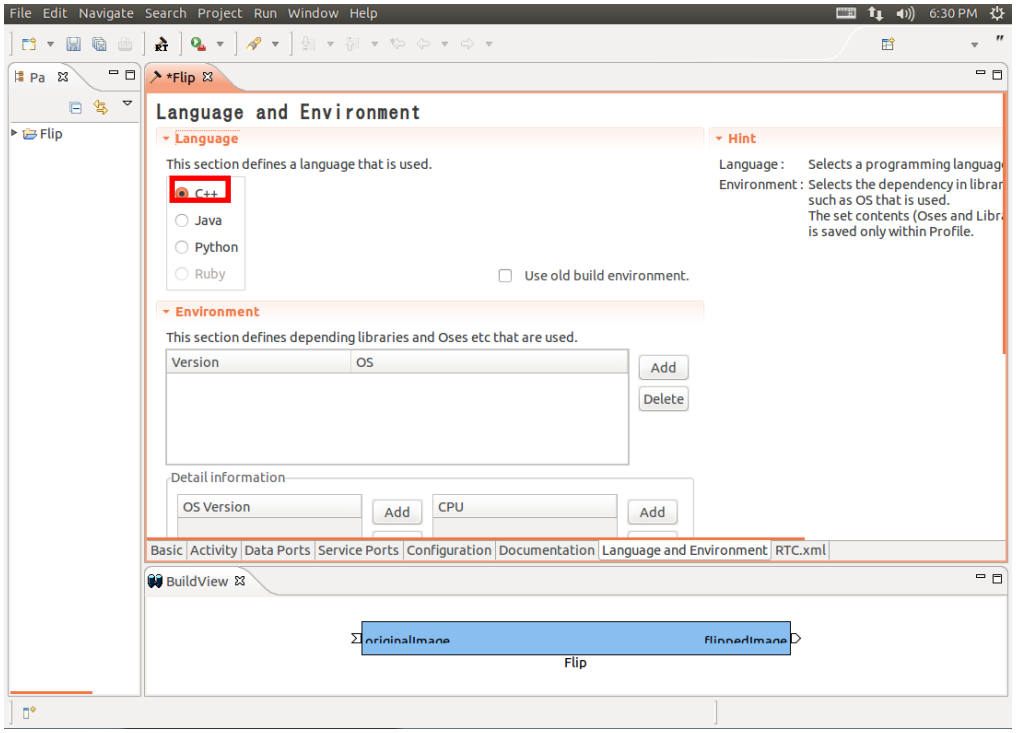


「Configuration」タブを選択し、先ほど決めた仕様を元に、Configuration の情報を入力します。制約条件および Widget とは、RTSystemEditor でコンポーネントのコンフィギュレーションパラメータを表示する際に GUI で値の変更を行うための形式を表すものです。
ここでは、flipMode の値は先ほど仕様を決めたときに、-1,0,1 の 3 つの値のみ取ることにしたので、ラジオボタンを使用することにします。

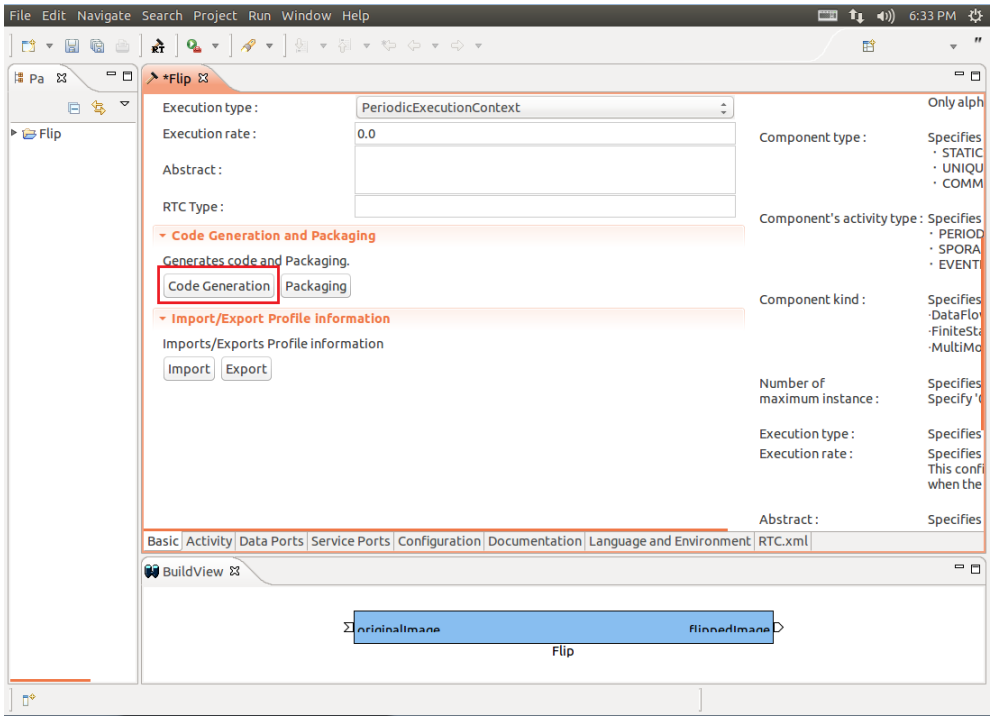
名称: flipMode
データ型: int
デフォルト値: 1
変数名: flipMode
制約条件: (-1, 0, 1)
Widget: radio



「Language and Environment」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。



全ての設定が完了しましたら、「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。



5. 仮ビルド

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。中身の実装は出来ていませんがこの状態でコンパイルは実行できます。

Linux 環境の場合 Flip コンポーネントのソースが生成されたディレクトリで下記コマンドをうちます。

```
$ cd rtcws/Flip
$ mkdir build
$ cd build
$ cmake ..
$ make
```

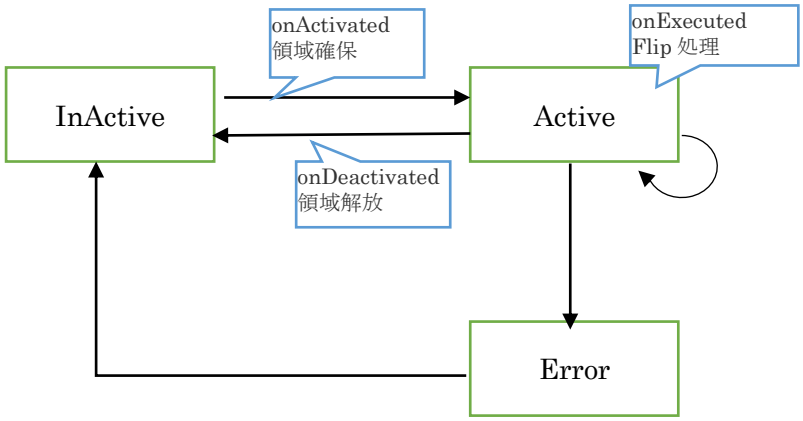
これで Configure およびビルドが完了します。問題なく出来ることを確認してください。

4. ヘッダ、ソースの編集

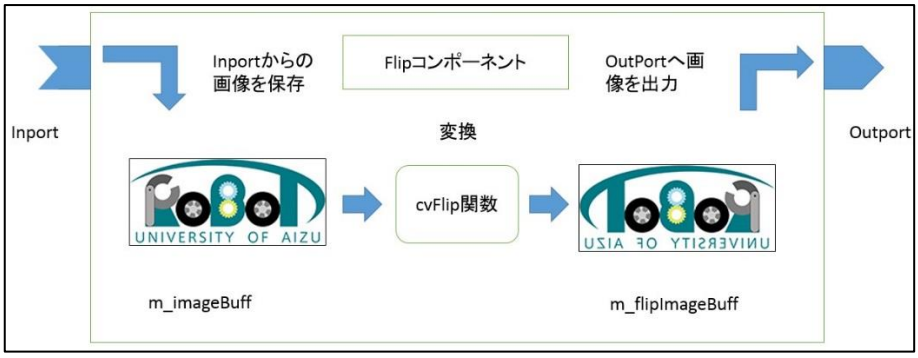
1. アクティビティ処理の実装

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()での処理内容は下記図になります。



Flip コンポーネントの処理の流れは以下の図になります。



2. ヘッダファイル（Flip.h）の編集

OpenCV のライブラリを使用するため、OpenCV のインクルード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 用インクルード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 の中に追加してください。

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

3. ソースファイル（Flip.cpp）の編集

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

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;
}
```

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;
}
```

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;
}
```

5. CMake によるビルドに必要なファイルの生成

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てやる必要が有ります。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。

```
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} ${OpenCV_LIBS}) ←　OepnCV_LIBS を追加
    : 中略
add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
   ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES} ${OpenCV_LIBS}) ←　OepnCV_LIBS を追加
```

6. ビルド

CMakeList.txt を編集したので、再度 CMake で Configure およびビルドを行います。

```
$ cd rtcws/Flip (eclipse ワークスペース下の Flip ディレクトリへ移動)
$ rm -rf build (念のため仮ビルドで作成したディレクトリは削除)
$ mkdir build (再度 build ディレクトリを作成)
$ cd build
$ cmake ..
$ make
```

エラーが出た場合は CMakeLists.txt やソースコードに間違えがないか確認してください。

7. Flip コンポーネントの動作確認

1. NameService の起動

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。

```
$ rtm-naming
```

(y/N)と聞かれたら y を選択してください。

2. Flip コンポーネントの起動

Flip コンポーネントを起動します下記コマンドで実行します。

```
$ cd rtcws/Flip/build/src (もし現在 build/src 以外にいる場合)
$ ./FlipComp
```

3. カメラコンポーネントとビューアコンポーネントの起動

USB カメラのキャプチャ画像を OutPort から出力する OpenCVCameraComp と InPort

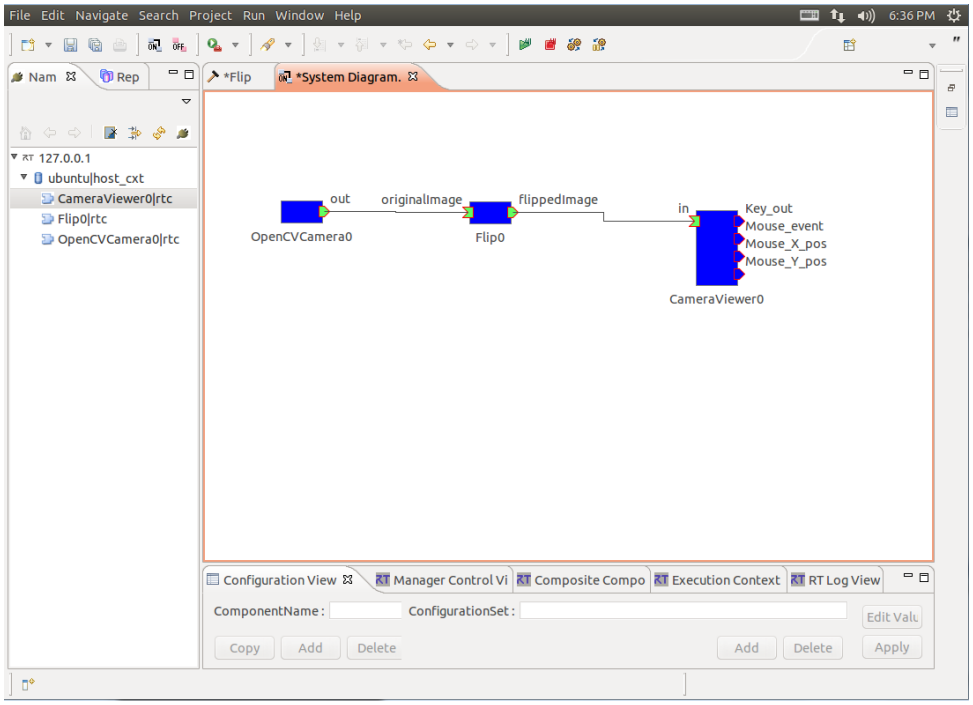
で受け取った画像を画面に表示する CameraViewerComp を起動します。

```
$ /usr/local/share/openrtm-1.1/components/c++/opencv-rtcs/OpenCVCameraComp
$ /usr/local/share/openrtm-1.1/components/c++/opencv-rtcs/CameraViewerComp
```

8. コンポーネントの接続

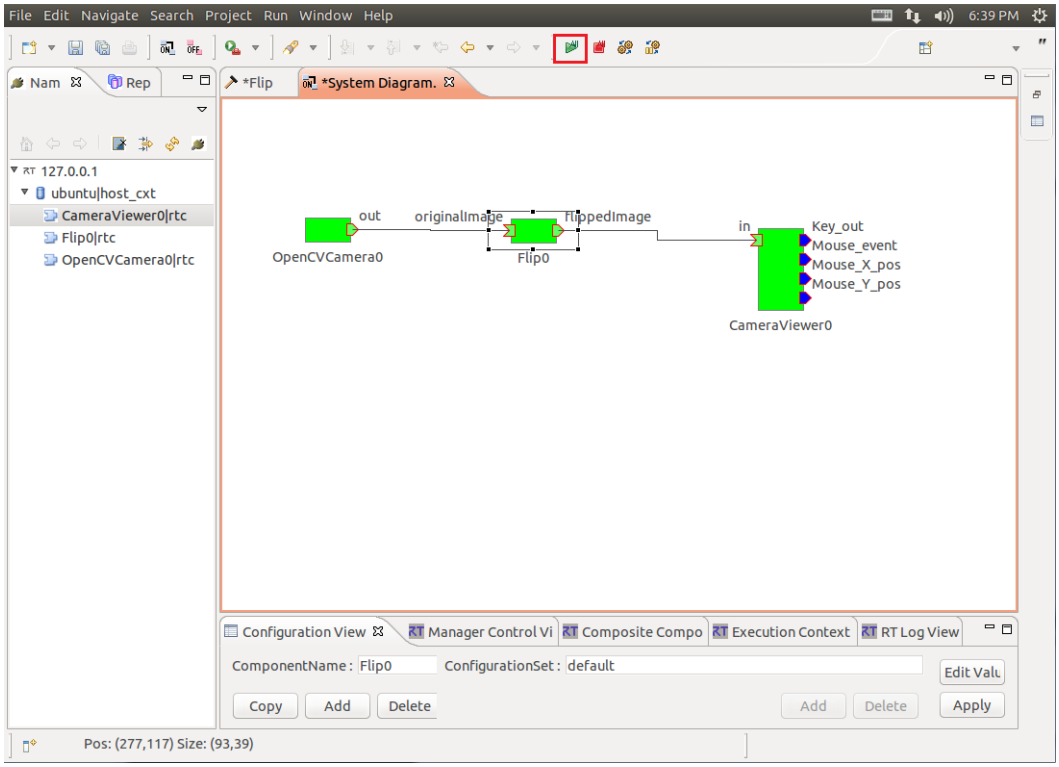
1. コンポーネントの接続

下記図の様に各コンポーネントを接続します。



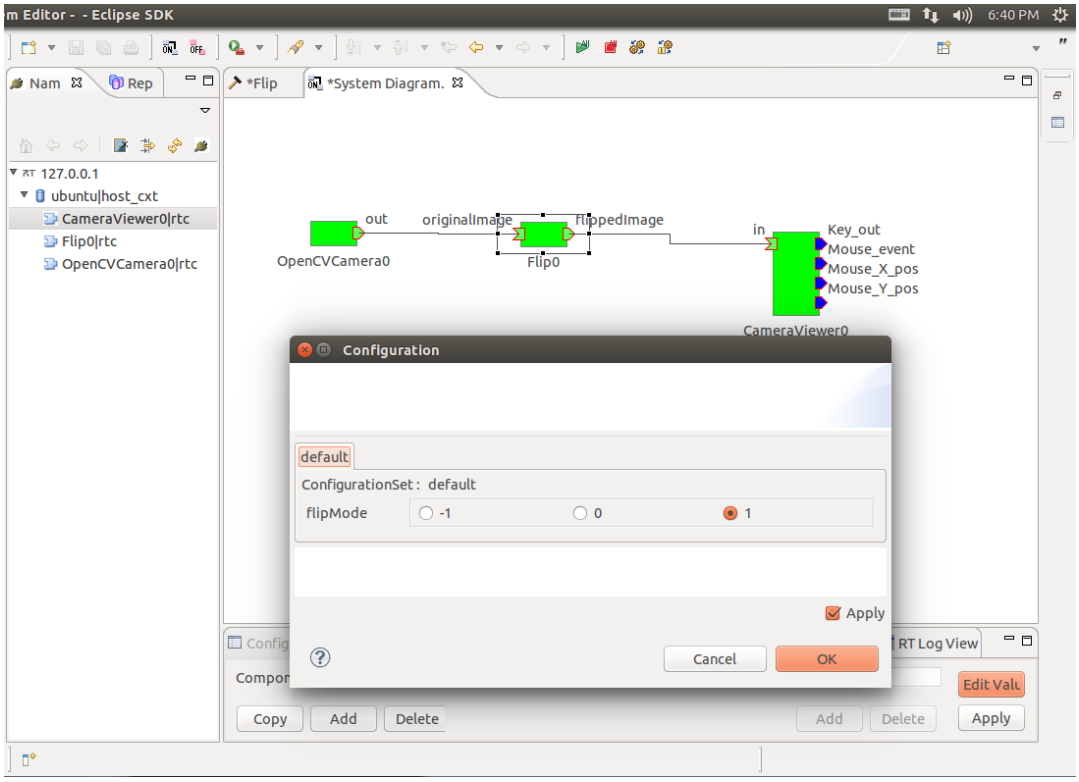
2. コンポーネントの Activate

RTSystemEditor の上部にあります「ALL」というアイコンをクリックし、 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。



3. 動作確認

下図のようにコンフィギュレーションビューにて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出てきます。「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。



3. RTC-Library-FUKUSHIMA

1. RTC-Library-FUKUSHIMA について

RTC-Library-FUKUSHIMA とはロボット産業振興のために作成された RTC ソフトウェアライブラリーです。

主にコンポーネントの登録やダウンロードしての再利用などが出来ます。

2. コンポーネントをアップロード

RTC-Library-FUKUSHIMA へのコンポーネントのアップロードの仕方を説明します。

1. ログイン

RTC-Library-FUKUSHIMA へは下記 URL でアクセスします。

RTC-Library-FUKUSHIMA：https://rtc-fukushima.jp/

今回の講習会では本番の環境を使わずにローカルの環境を使用します。

サイトにアクセス出来たらサイト上部のログインをクリックしてください。
ログイン画面に移行しユーザー名またはメールアドレスとパスワードを入力する欄がありますので入力してログインをしてください。

2. コンポーネントのアップロード手順
コンポーネントをアップロードするにはログイン後、トップページから「ライブラリ」を選択し、「コンポーネント登録/パッケージ登録」のタブを選択します。そして「コンポーネント登録」を選択します。

下記登録画面に遷移したことを確認してください。下記画面で登録を行います。

今回は下記項目を登録します。

RTC.xml ファイル読み込み
Flip コンポーネントで作成された RTC.xml を指定します。指定後、「RTC.xml ファイル読み込み」のボタンを押してください。
RTCBuilder で設定したコンポーネントの情報が登録されます。

コンポーネント登録情報入力

・コンポーネント名：Flip
・概要: Flip component
・カテゴリ：カメラ
・タグ：C++、OpenCV、画像処理
・ファイルアップロード：コンポーネントを zip に圧縮してアップロードします。その際、build 以下は削除か退避しておいてください。
・同意する：チェックを入れてください。
・私はロボットではありません：チェックを入れてください。※ローカル環境ではなし

入力が終わりましたら、「確認」のボタンを押し登録情報確認ページに遷移してください。

4. Flip コンポーネントの全ソース

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：https://rtc-fukushima.jp/wp/wp-content/uploads/2016/02/Flip_cpp.txt

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：https://rtc-fukushima.jp/wp/wp-content/uploads/2016/02/Flip_h.txt

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：https://rtc-fukushima.jp/wp/wp-content/uploads/2016/02/Flip.zip