

画像処理コンポーネントの作成

(独)産業技術総合研究所

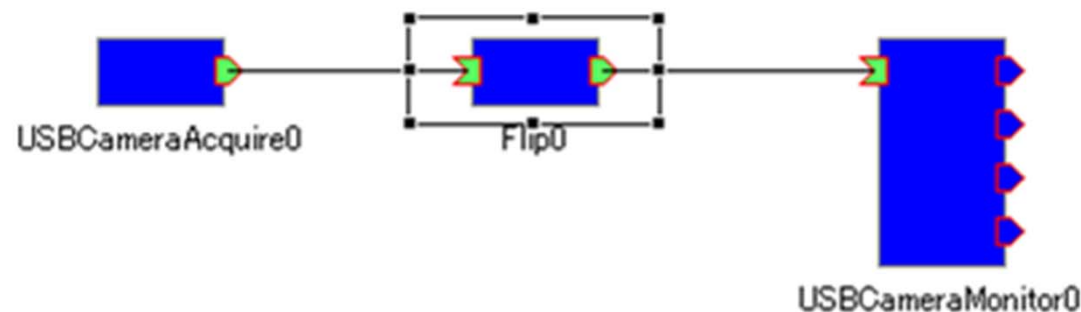
知能システム研究部門

宮本晴美



この実習では...

- デモストレーションで使用したFlipコンポーネントを、インストールしたソフトウェア・ツールを用いて、一から開発します。



コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- **OpenCVとcvFlip関数についての確認**
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- **コンポーネントの仕様を決める**
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- **CMakeを使ったプロジェクトの作成**
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- **ヘッダ、ソースの編集**
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- **コンポーネントの動作確認**

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

動作環境・開発環境についての確認

- OS: Windows XP SP3 (Vista, 7でも可能)
- コンパイラ: Visual C++ 2010 Express Edition 日本語版
- OpenRTM-aist-1.1.0-RELEASE (C++版), Win32 VC2010
- OpenRTP 1.1.0-RC4
 - RTSystemEditor 1.1
 - RTCBuilder 1.1
- Eclipse 3.8.1 (OpenRTP 1.1.0-RC4) Windows用全部入り
- Doxygen ドキュメント生成に必要
- CMake

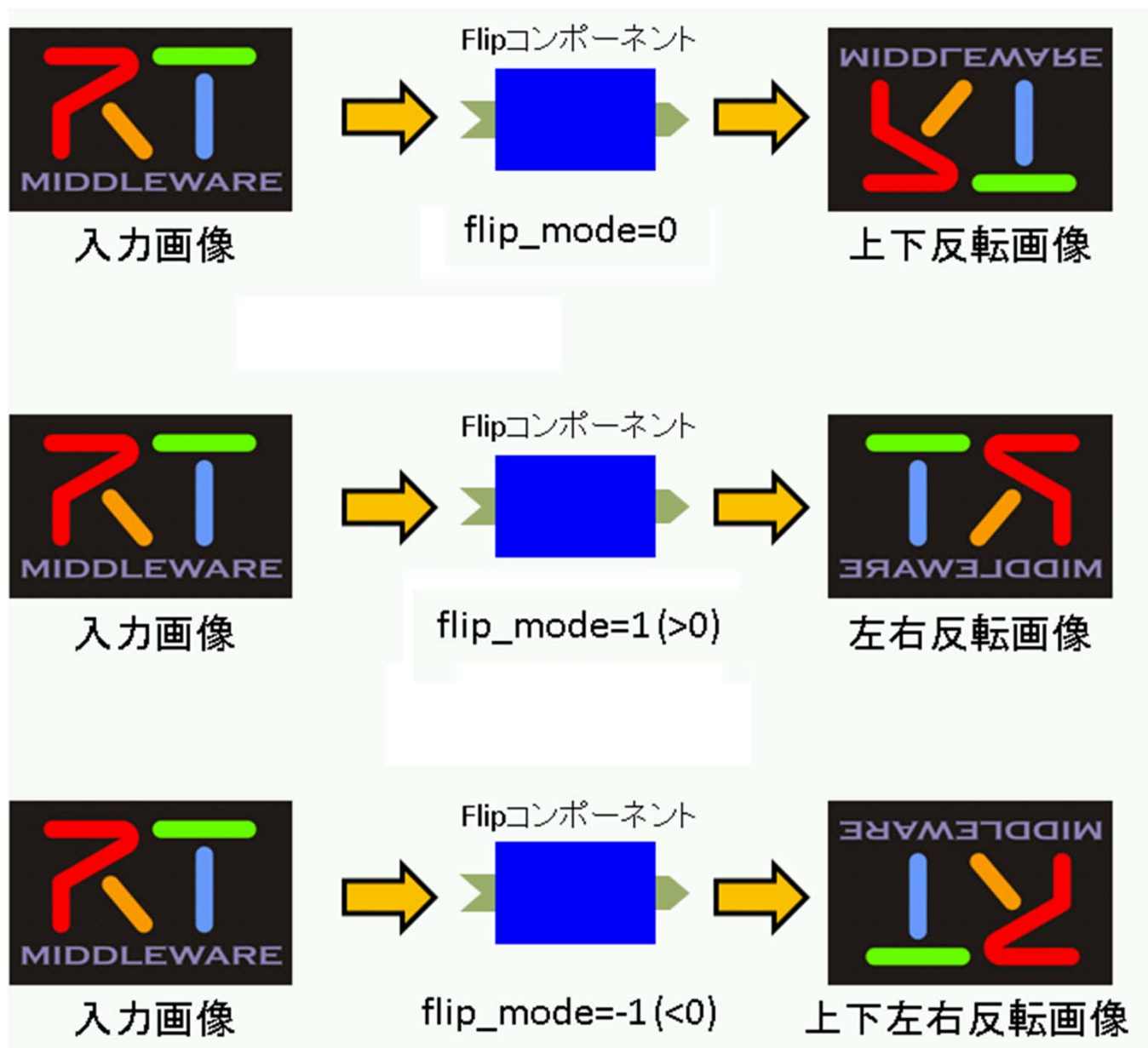
コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

OpenCVとcvFlip関数についての確認

```
void cvFlip(IplImage* src,  
            IplImage* dst=NULL,  
            int flipMode=0);
```

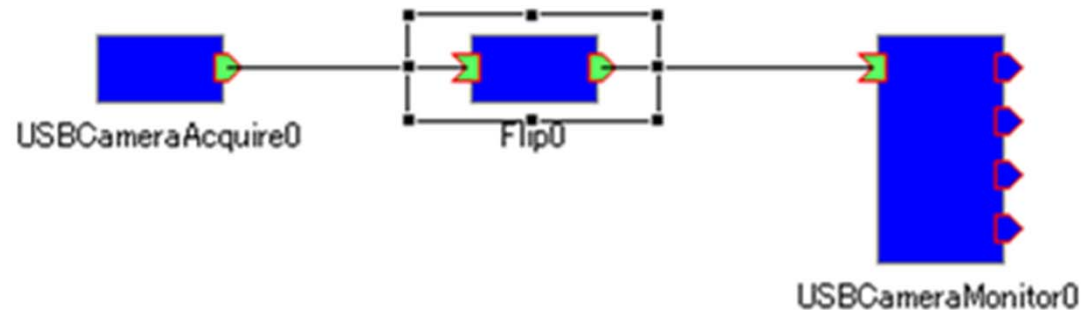
- src 入力配列
- dst 出力配列。もしdst=NULLであれば、srcが上書きされます。
- flipMode 配列の反転方法の指定内容:
 - flipMode = 0: X軸周りでの反転(上下反転)
 - flipMode > 0: Y軸周りでの反転(左右反転)
 - flipMode < 0: 両軸周りでの反転(上下左右反転)



コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- **コンポーネントの仕様を決める**
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

データポートの仕様



- OpenRTM-aistには OpenCVを使用したビジョン関連のコンポーネントがサンプルとして付属しています。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以下のような **CameraImage 型** を使用しています。
- 詳細はwebページを確認してください。

```

struct CameraImage {
    /// Time stamp.
    Time tm;
    /// Image pixel width.
    unsigned short width;
    /// Image pixel height.
    unsigned short height;
    /// Bits per pixel.
    unsigned short bpp;
    /// Image format (e.g. bitmap, jpeg, etc.).
    string format;
    /// Scale factor for images, such as disparity maps,
    /// where the integer pixel value should be divided
    /// by this factor to get the real pixel value.
    double fDiv;
    /// Raw pixel data.
    sequence<octet> pixels;
};

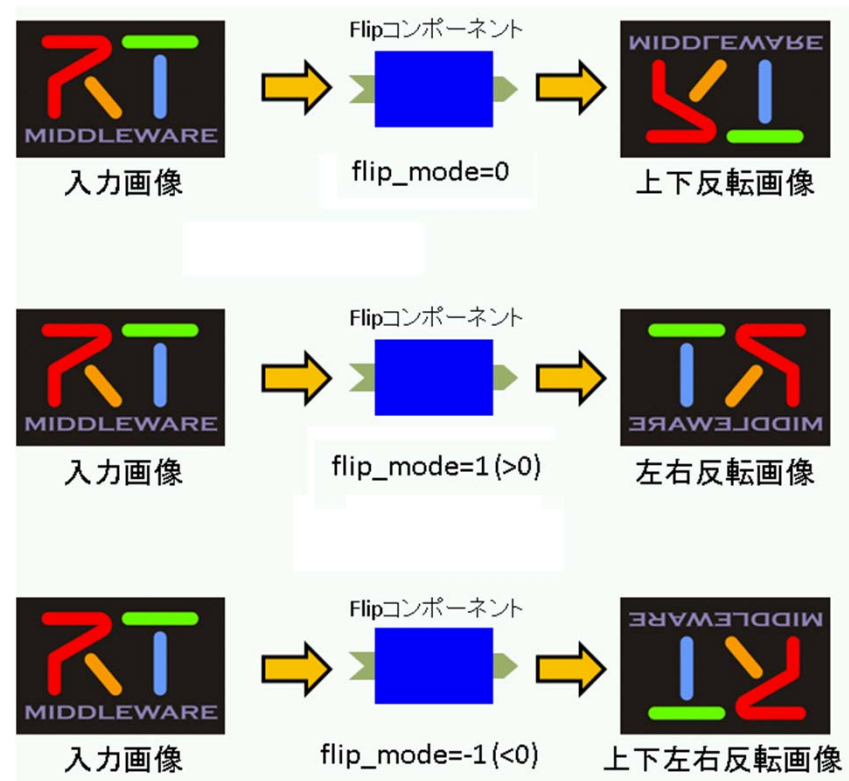
```

コンフィギュレーションの仕様

```
void cvFlip(IplImage* src,  
            IplImage* dst=NULL,  
            int flipMode=0);
```

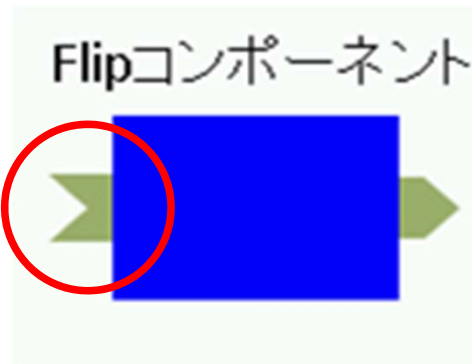
- src 入力配列
- dst 出力配列。もしdst=NULLであれば、srcが上書きされます。
- flipMode 配列の反転方法の指定内容:
 - flipMode = 0: X軸周りでの反転(上下反転)
 - flipMode > 0: Y軸周りでの反転(左右反転)
 - flipMode < 0: 両軸周りでの反転(上下左右反転)

コンフィギュレーションの仕様



- ひとつのコンポーネントで3つの変換ができると、ユーザーも便利だし、管理するほうも楽チンですよ。

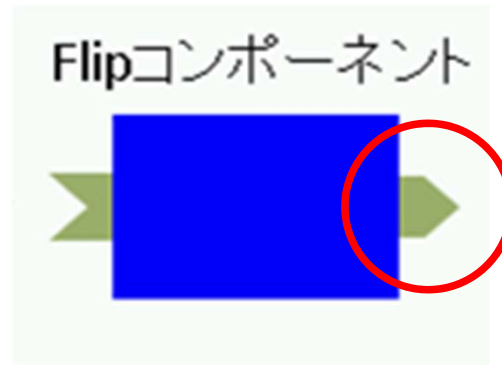
仕様まとめ



● InPort

- ポート名 **originalImage**
- 型 CameraImage
- 意味 入力画像

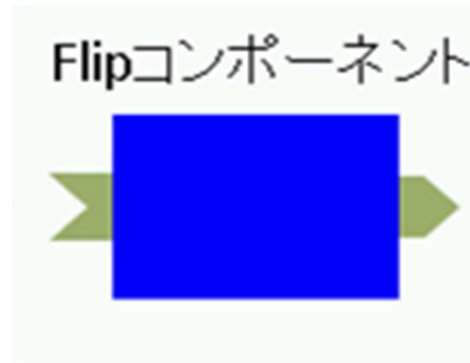
仕様まとめ



● OutPort

- ポート名 **flippedImage**
- 型 CameraImage
- 意味 反転された画像

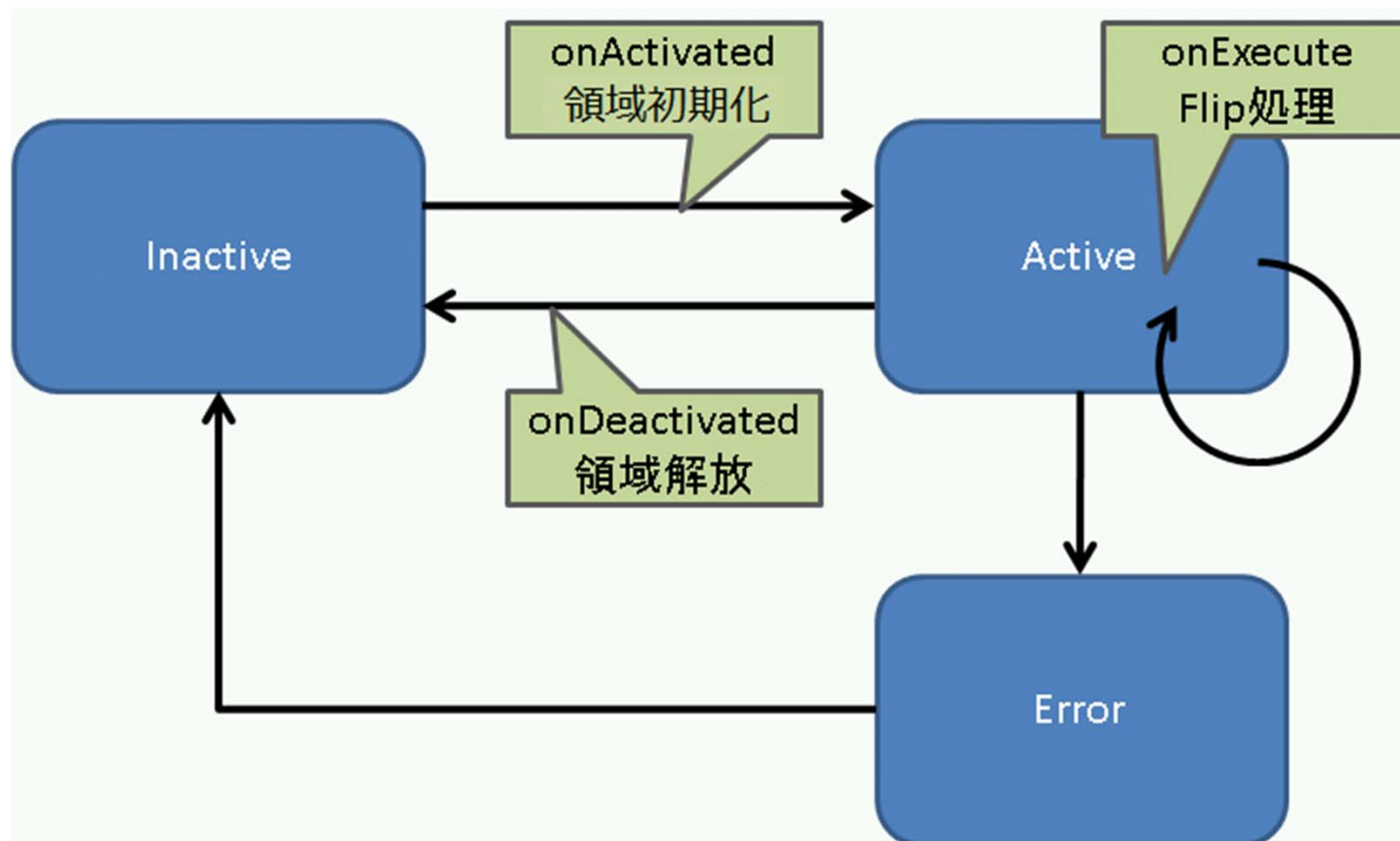
仕様まとめ



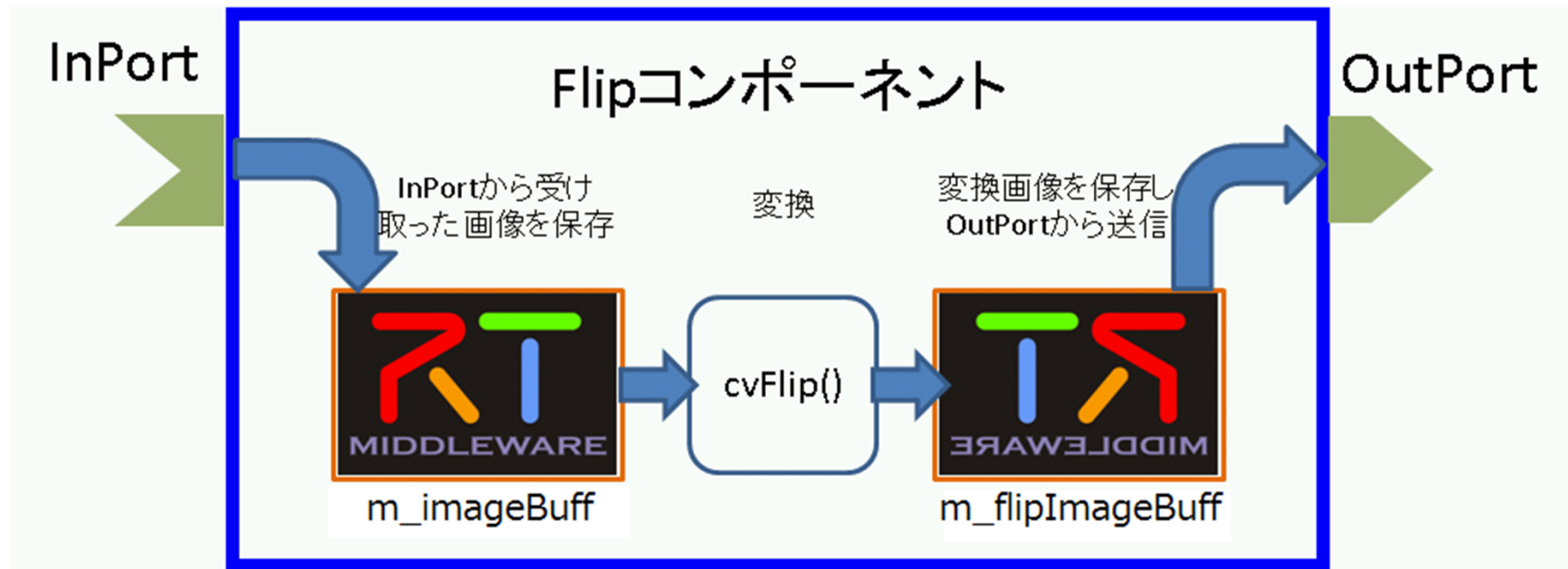
- Configuration

- 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

アクティビティ処理の実装



アクティビティ処理の実装



コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

RTCBuilderを用いた ソースコードのひな形の作成・手順

- eclipseを起動し、RTCBuilderを表示する
- 新規プロジェクトの作成
- コンポーネント情報を入力し生成する
 - XMLをインポートする、または、情報を入力する。

※この章の実習は、初めてEclipseを起動した場合、または新しいワークスペースを使用した場合を想定しております。一緒に実習をする方は、新しいワークスペースを使用してください。



入力に必要なコンポーネントの詳細情報は
Webページをご覧ください。
最後に生成されたファイルを確認します。

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- **CMakeを使ったプロジェクトの作成**
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

CMakeを使ったプロジェクトの作成

- CMakeを起動し、ディレクトリを設定する
- Configureボタンを押下し、コンフィギュアする。
- Generate ボタンを押下し、プロジェクトを生成する。

※この章の実習も、初めてCMakeを起動した場合、を想定しております。

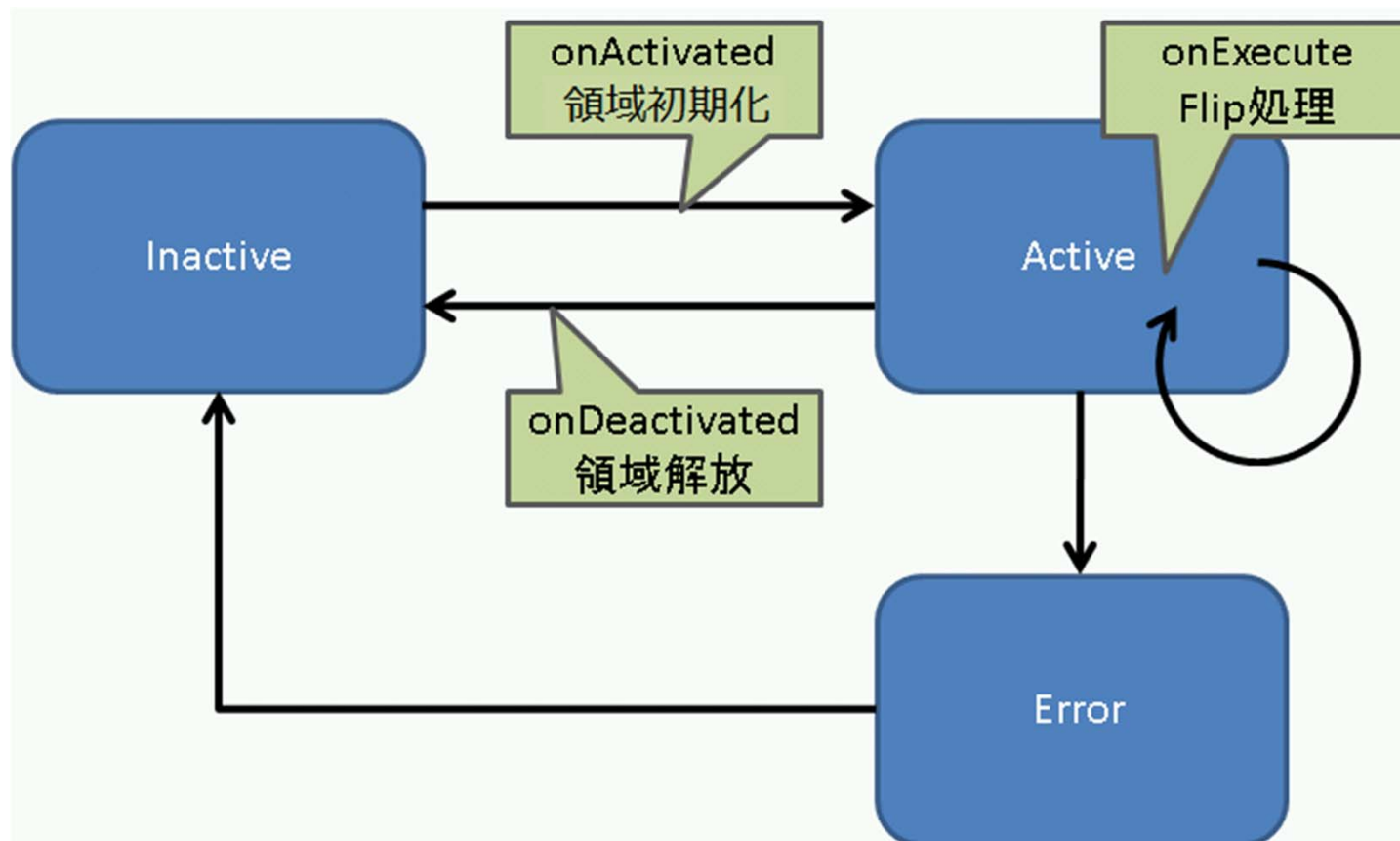
最後に生成されたプロジェクトを
VC++でビルドして確認します。



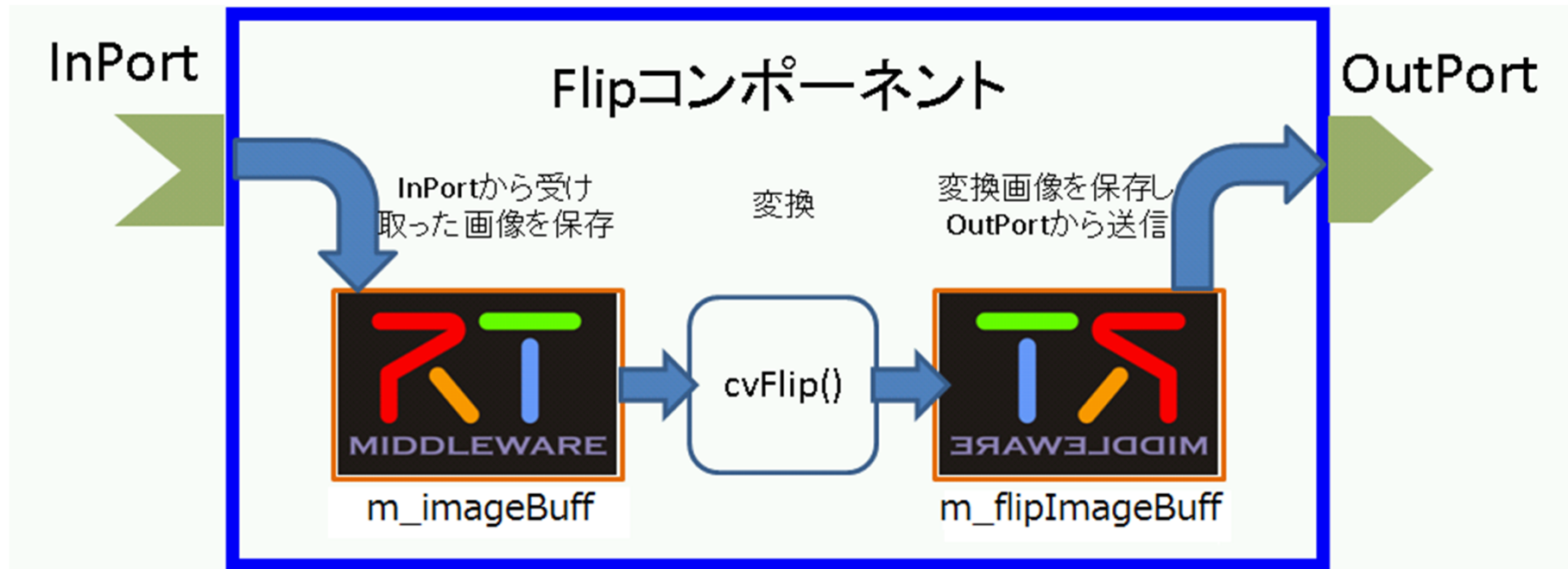
コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- **ヘッダ、ソースの編集**
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

ヘッダ、ソースの編集



ヘッダ、ソースの編集



ヘッダの編集

- OpenCVのインクルード文を追加する。
- イメージ用メモリ変数を宣言する。
 - m_imageBuff
 - m_flipImageBuff

入力に必要なコンポーネントの詳細情報は
Webページをご覧ください。

ソースの編集

- onActivated()
 - イメージ用メモリの初期化と、outportの画面サイズの初期化を行います。
- onDeactivated()
 - イメージ用メモリを開放します。
- onExecute()
 - インポートの確認・読み込み、vcFlip関数の処理、アウトポートのデータ送信などを行います。
入力に必要なコンポーネントの詳細情報は
Webページをご覧ください。

コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- コンポーネントの動作確認

CMakeを使ったプロジェクトの変更

- OpenCVを使う為に、インクルード文のほかに次の設定が必要です。
 - インクルードパス
 - ライブラリ
 - ライブラリパス
- CMakeLists.txtを修正して、OpenCVの設定が入ったCMakeを実行しプロジェクトを再生成します。
- 完了したらVC++でビルドします。

修正内容はWebページをご覧ください。

ではCMakeを修正します。

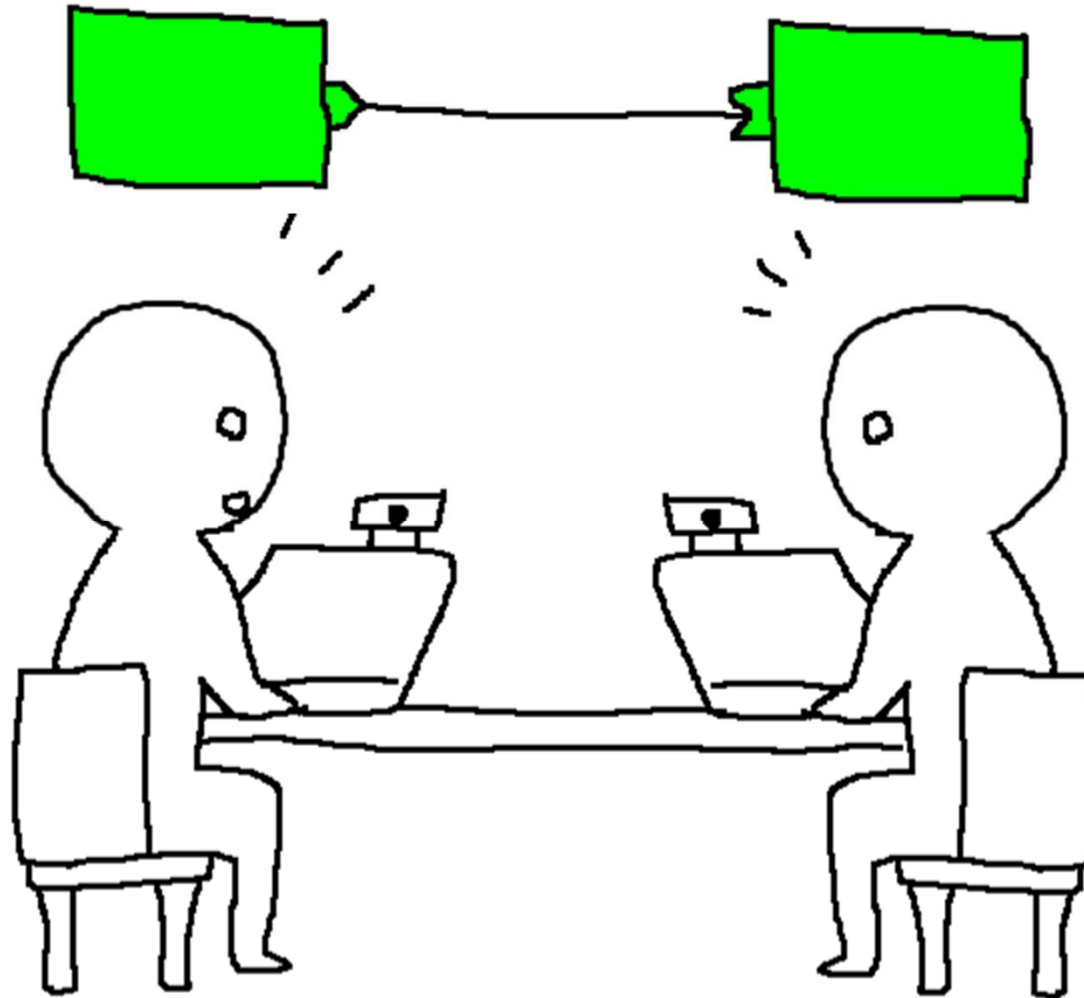
コンポーネント作成手順

- 動作環境・開発環境についての確認
- OpenCVとcvFlip関数についての確認
- コンポーネントの仕様を決める
- RTCBuilderを用いたソースコードのひな形の作成
- CMakeを使ったプロジェクトの作成
- ヘッダ、ソースの編集
- CMakeを使ったプロジェクトの変更
- **コンポーネントの動作確認**

コンポーネントの動作確認

- NameServiceの起動
- Flipコンポーネントの起動
- その他のコンポーネントの起動
 - OpenCVCameraComp, またはDirectShowCamComp
 - CameraViewerComp
- RTSystemEditorでコンポーネントを確認・接続する
- コンポーネントをActivateし、動作確認を行う

隣の人とのコンポーネントと通信する



隣の人のコンポーネントと通信する

- 操作手順
 - rtc.confを作成する
 - ネームサービス、RTコンポーネント、を起動する
 - RTSystemEditorを起動する
 - ネームサービスビューに、隣の人のネームサービスを登録する
 - システムダイアログに表示し、データポートをリンクする

隣の人とのコンポーネントと通信する

● 操作手順

- ネットワークを確認し、rtc.confを作成する
- RTコンポーネントを再起動する
- RTSystemEditorを起動する
- ネームサービスビューに、隣の人のネームサービスを登録する
- システムダイアログに表示し、データポートをリンクする

隣の人のコンポーネントと通信する

- ネットワークの確認

- 無線LANは通信量が増えると通信速度が遅くなることがあるため、有線LANを使います。

- rtc.confの準備

- 有線LANと無線LANを使用すると、ネットワークインタフェースが二つになります。rtc.confを作成し、コルバエンドポイントを設定します。

- rtc.conf

- corba.nameservers: localhost

- naming.formats: %n.rtc

- corba.endpoints: 192.168.XXX.XXX

隣の人のコンポーネントと通信する

- ネットワークの確認

- 無線LANは、通信量が増えると通信速度が遅くなることがあるため、有線LANを使います。

- rtc.confの準備

- 有線LANと無線LANを使用すると、ネットワークインタフェースが二つになります。rtc.confを作成し、コルバエンドポイントを設定します。

- **rtc.conf**

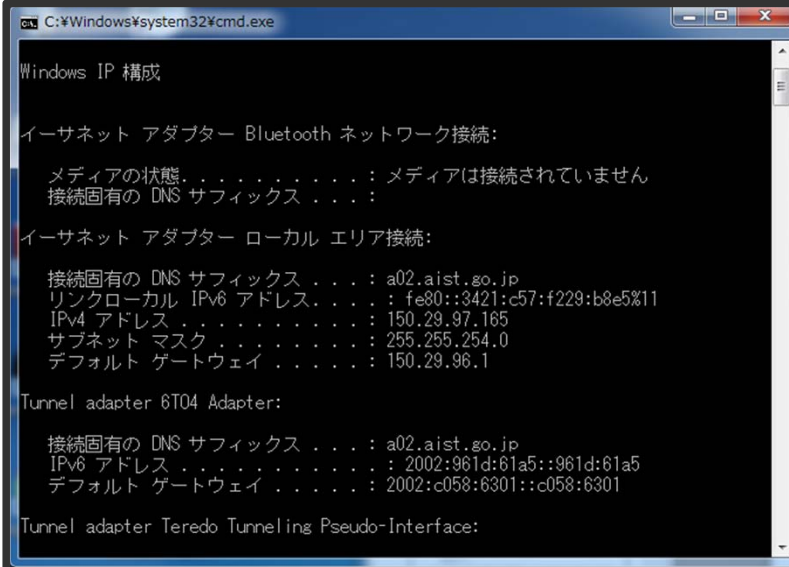
- corba.nameservers: localhost
- naming.formats: %n.rtc

- corba.endpoints: localhost, 192.168.XXX.XXX:

IPアドレスの確認方法

- コマンドプロンプトで確認する

- 「スタートボタン」の「プログラムとファイルの検索」に「cmd.exe」と入力し「Enter」を押下し、コマンドプロンプトを表示します。
- コマンドプロンプトに「ipconfig」と入力しIPアドレスを確認します。



```
C:\Windows\system32\cmd.exe
Windows IP 構成

イーサネット アダプター Bluetooth ネットワーク接続:
   メディアの状態 . . . . . : メディアは接続されていません
   接続固有の DNS サフィックス . . . . . :

イーサネット アダプター ローカル エリア接続:
   接続固有の DNS サフィックス . . . . . : a02.aist.go.jp
   リンクローカル IPv6 アドレス . . . . . : fe80::3421:c57:f229:b8e5%11
   IPv4 アドレス . . . . . : 150.29.97.165
   サブネット マスク . . . . . : 255.255.254.0
   デフォルト ゲートウェイ . . . . . : 150.29.96.1

Tunnel adapter 6T04 Adapter:
   接続固有の DNS サフィックス . . . . . : a02.aist.go.jp
   IPv6 アドレス . . . . . : 2002:961d:61a5::961d:61a5
   デフォルト ゲートウェイ . . . . . : 2002:c058:6301::c058:6301

Tunnel adapter Teredo Tunneling Pseudo-Interface:
```


IPアドレスの確認方法

- コマンドプロンプトで確認する(続き)
 - 「192.168.XXX.XXX」を使用します。有線LANで接続したルータのDHCP機能で配布されたアドレスです。
 - ちなみに「150.29.97.XXX」は産総研の無線LANで配布されたアドレスでインターネットに接続できません。

rtc.conf

- corba.nameservers: localhost
- naming.format: %n/rtc
- corba.endpoints: localhost, 192.168.XXX.XXX:

隣の人とのコンポーネントと通信する

● 操作手順

- ネットワークを確認し、rtc.confを作成する
- RTコンポーネントを再起動する
- RTSystemEditorを起動する
- ネームサービスビューに、隣の人のネームサービスを登録する
- システムダイアグラムに表示し、データポートをリンクする

隣の人のコンポーネントと通信する

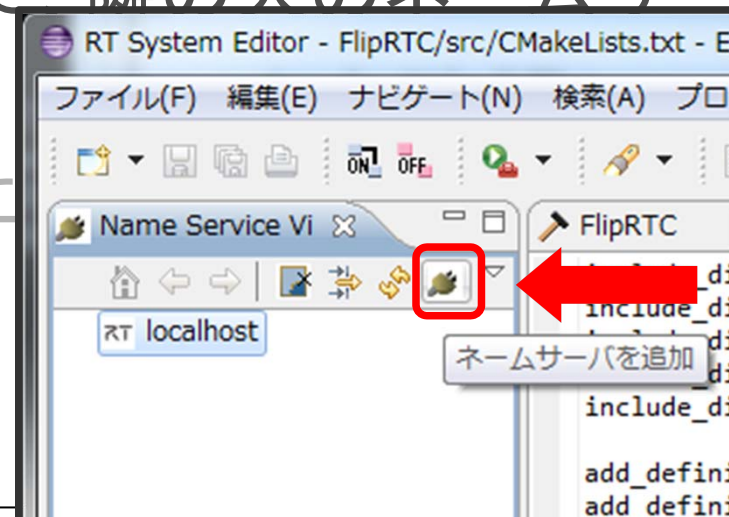
● 操作手順

- ネットワークを確認し、rtc.confを作成する
- RTコンポーネントを再起動する
- **RTSystemEditorを起動する**
- ネームサービスビューに、隣の人のネームサービスを登録する
- システムダイアグラムに表示し、データポートをリンクする

隣の人とのコンポーネントと通信する

● 操作手順

- ネットワークを確認し、rtc.confを作成する
- RTコンポーネントを再起動する
- RTSystemEditorを起動する
- ネームサービスビューに 隣の人のネームサービスを登録する
- システムダイアグラムにリンクする



隣の人とのコンポーネントと通信する

● 操作手順

- ネットワークを確認し、rtc.confを作成する
- RTコンポーネントを再起動する
- RTSystemEditorを起動する
- ネームサービスビューに、隣の人のネームサービスを登録する
- システムダイアグラムに表示し、データポートをリンクする

☆お疲れ様でした☆