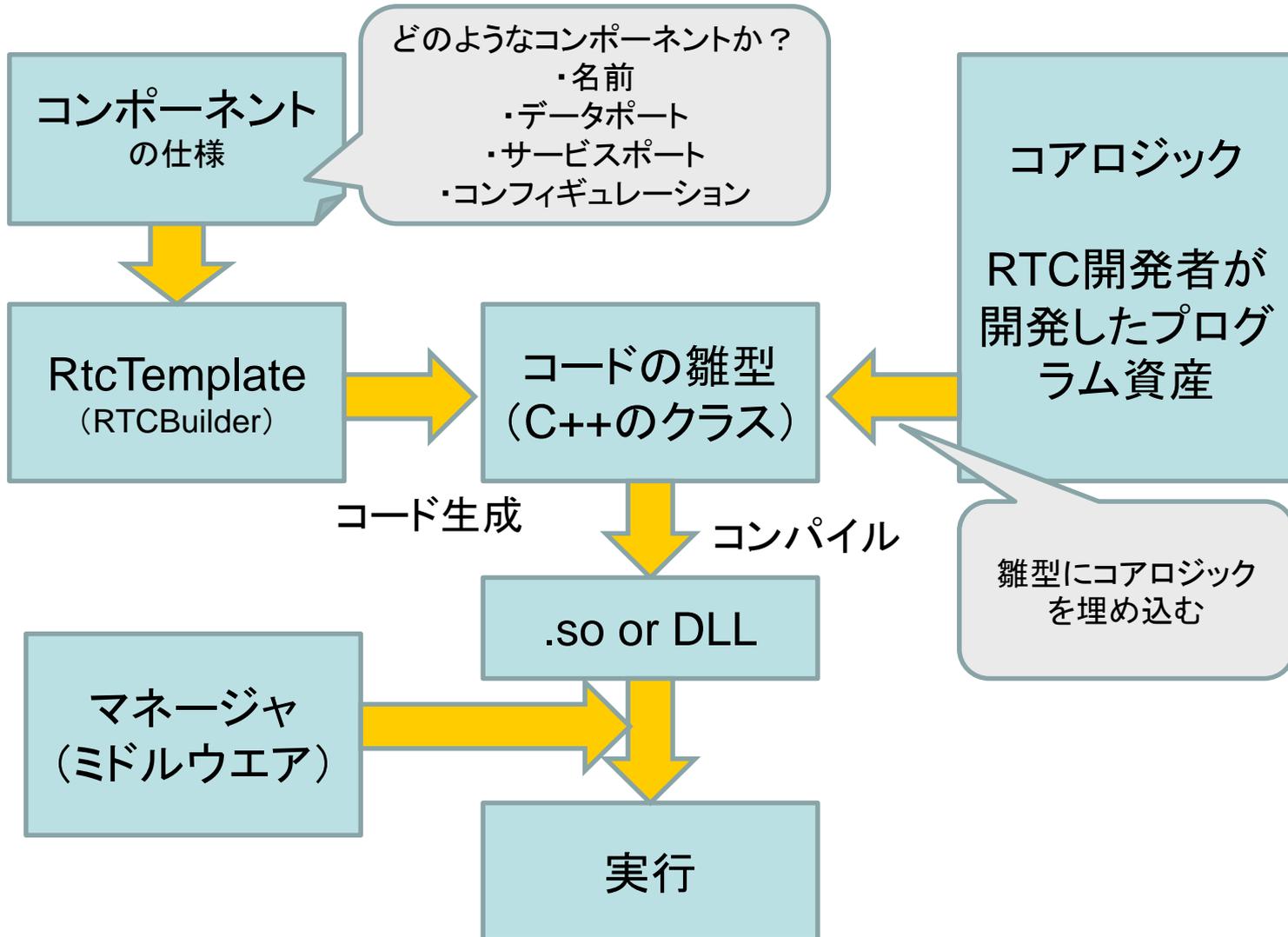


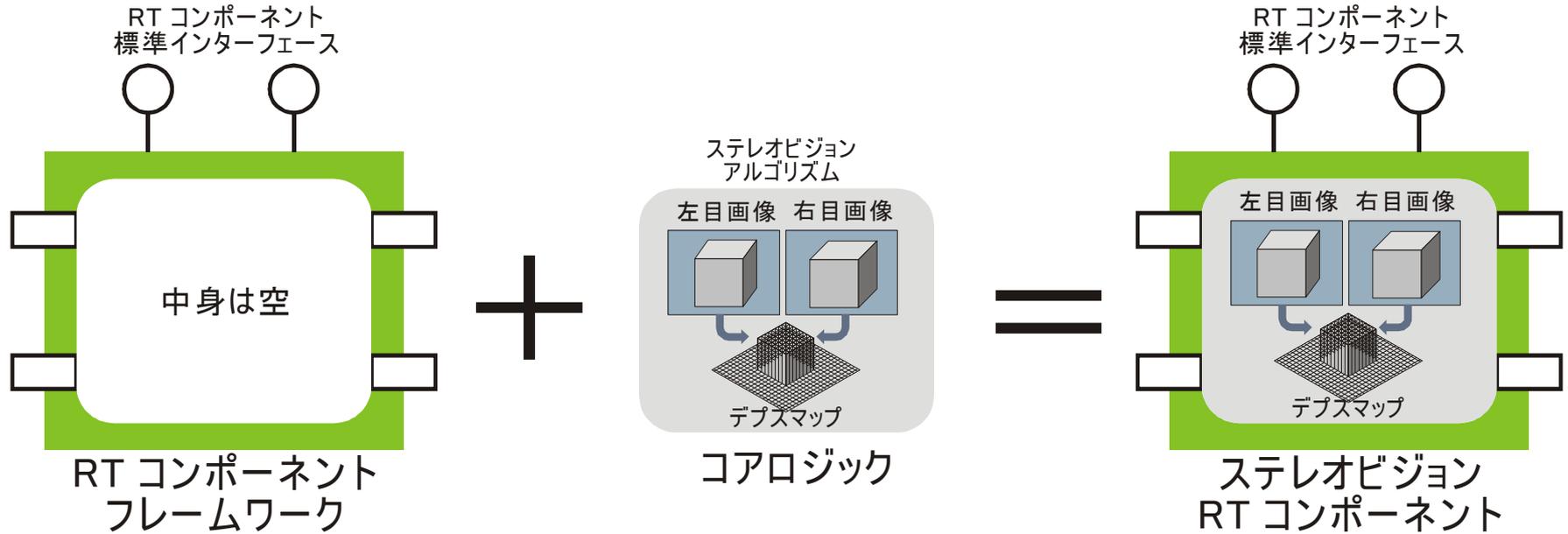
第2部 RTコンポーネントの開発



OpenRTMを使った開発の流れ

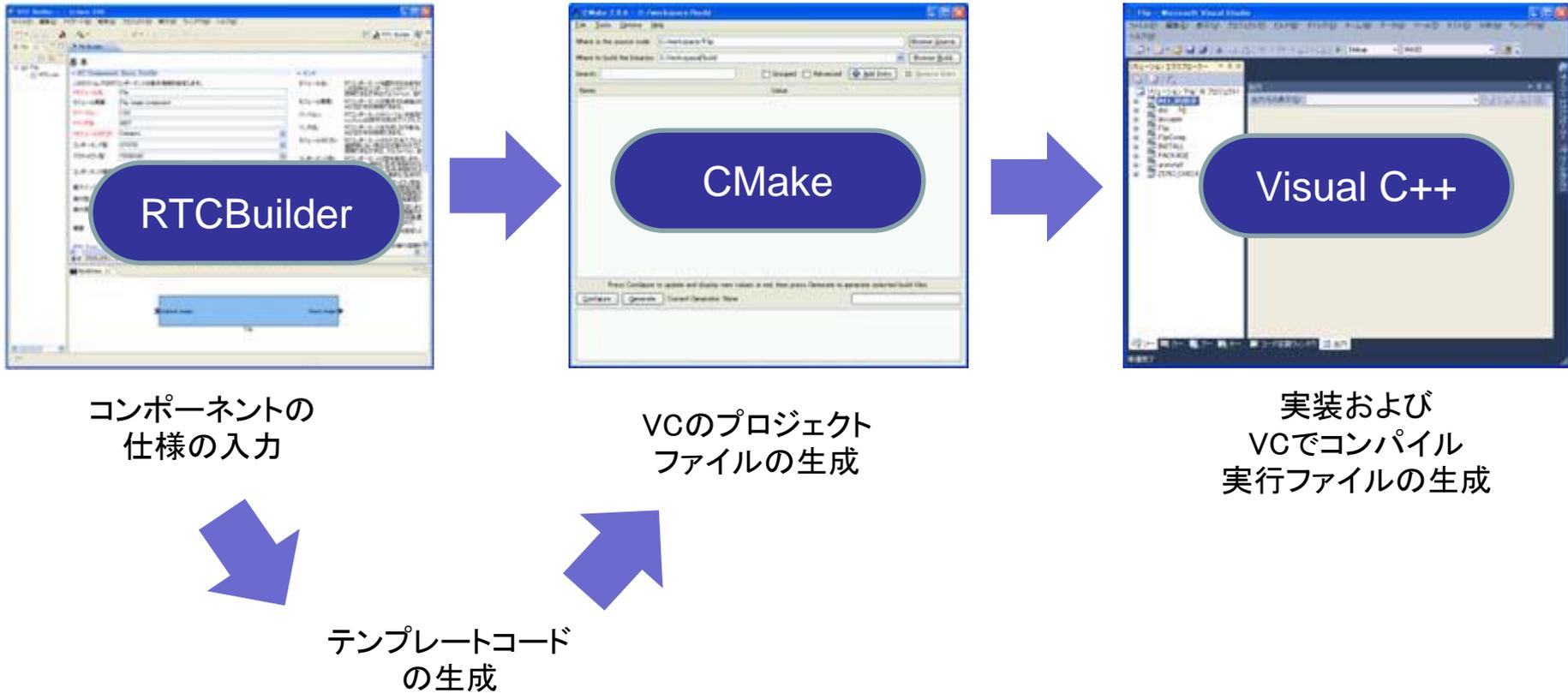


フレームワークとコアロジック



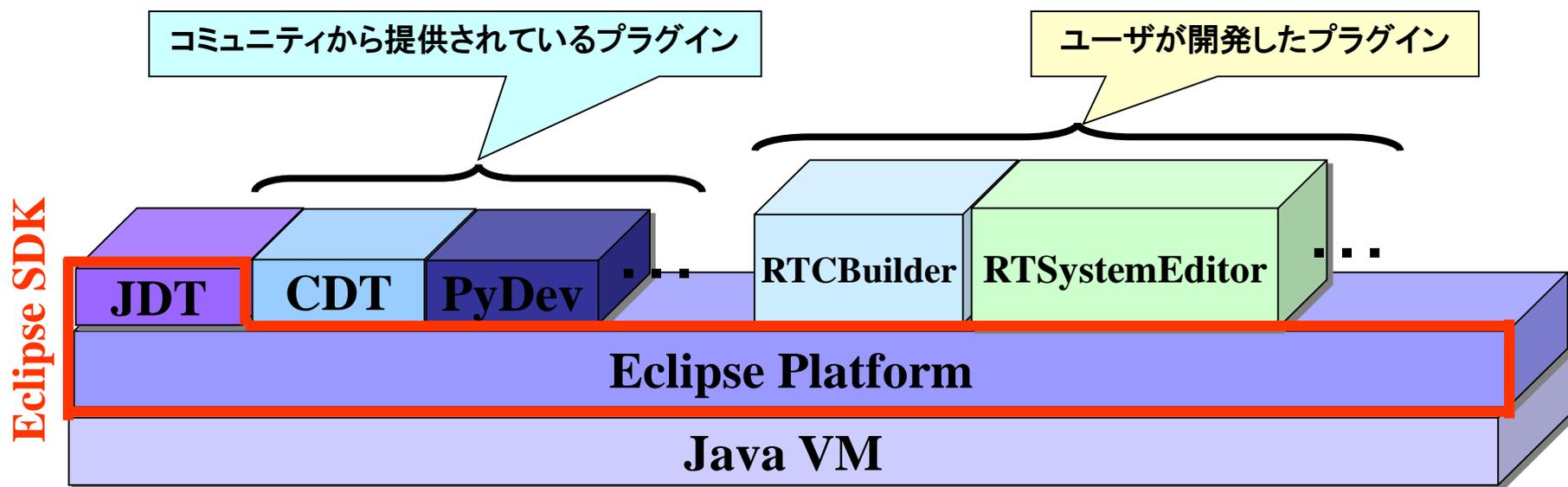
RTCフレームワーク+コアロジック=RTコンポーネント

コンポーネントの作成 (Windowsの場合)



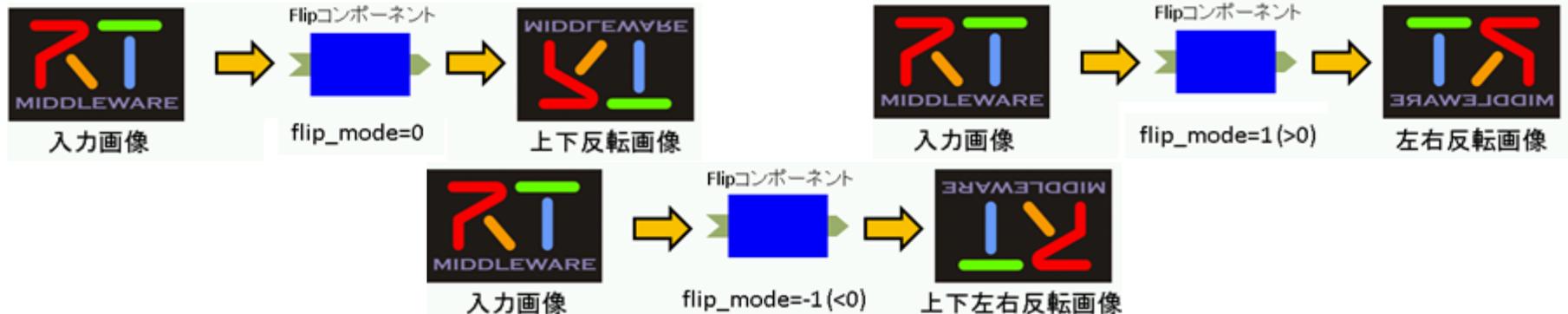
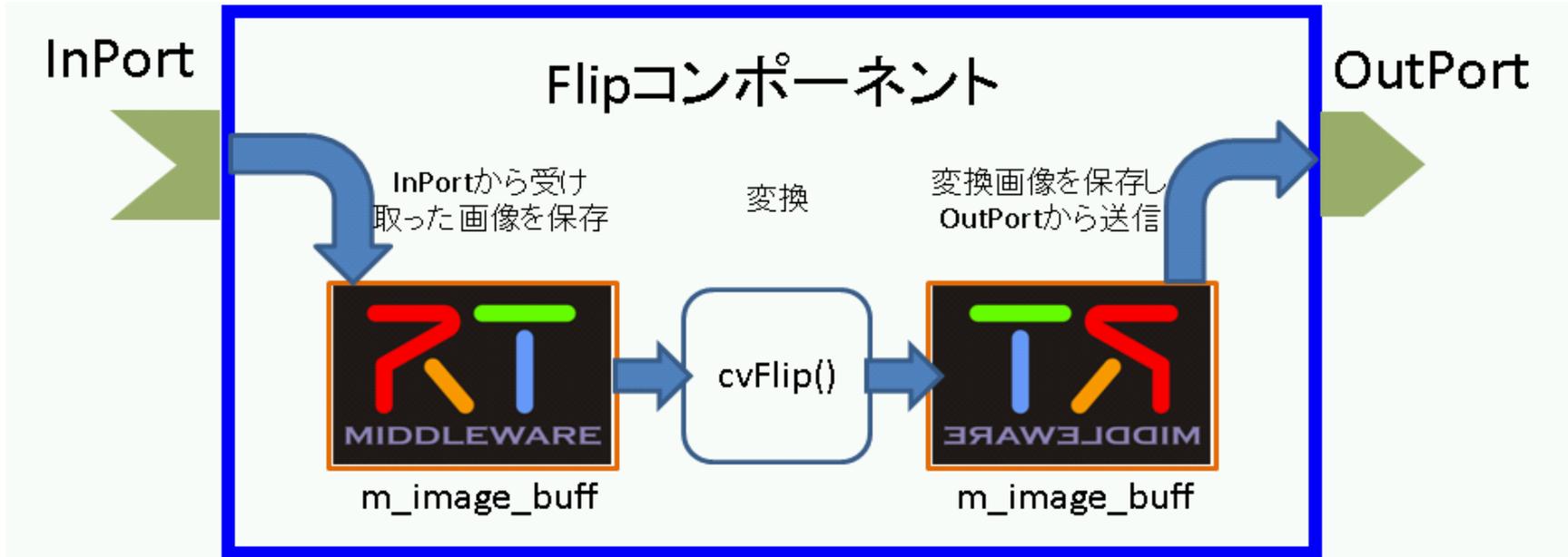
統合開発環境Eclipse

- オープンソース・コミュニティで開発されている統合開発環境
 - ◆ マルチプラットフォーム対応. WindowsやLinuxなど複数OS上で利用可能
 - ◆ 「Plug-in」形式を採用しており, 新たなツールの追加, 機能のカスタマイズが可能
 - ◆ RCP(Rich Client Platform)を利用することで, 簡単に単独アプリ化が可能



Flipコンポーネントについて

- 入力画像を反転して出力するコンポーネント
 - OpenCVのcvFlip関数を利用



システム構築支援ツール RTSystemEditorについて

- RTSystemEditorとは？
 - RTコンポーネントを組み合わせて、RTシステムを構築するためのツール

The screenshot shows the RTSystemEditor interface with several views highlighted by colored boxes and labels:

- システムエディタ** (System Editor): The central workspace showing a diagram with components like `MyServiceProvider0` and `ConfigSample0`.
- 名前サービスビュー** (Name Service View): A view on the left showing a tree structure of components and services.
- プロパティビュー** (Property View): A view on the right showing the configuration properties for a selected component.
- コンフィギュレーションビュー** (Configuration View): A view at the bottom showing a table of configuration parameters.
- マネージャビュー** (Manager View): A view at the bottom left showing loaded modules and active components.
- 複合コンポーネントビュー** (Composite Component View): A view at the bottom center showing a table of composite components.
- 実行コンテキストビュー** (Execution Context View): A view at the bottom right showing execution context details.
- ログビュー** (Log View): A view at the bottom right showing a log of system events.

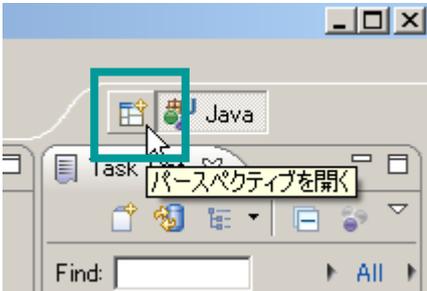
- Naming Serviceの起動
 - [スタート]メニューから
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]

- CameraViewerCompの起動
 - [スタート]メニューから起動
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
 - [opencv-rtcs]→ [CameraViewerComp.exe]

- DirectShowCamCompの起動
 - [スタート]メニューから起動
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
 - [opencv-rtcs]→ [DirectShowCamComp.exe]

n パースペクティブの切り替え

①画面右上の「パースペクティブを開く」を選択し、一覧から「その他」を選択



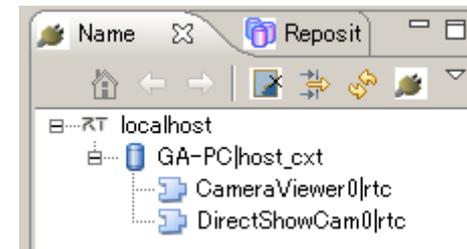
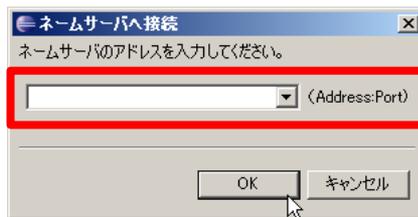
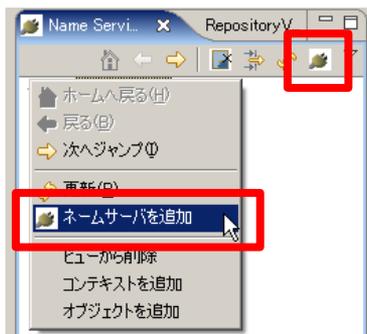
②一覧画面から対象ツールを選択



※ パースペクティブ

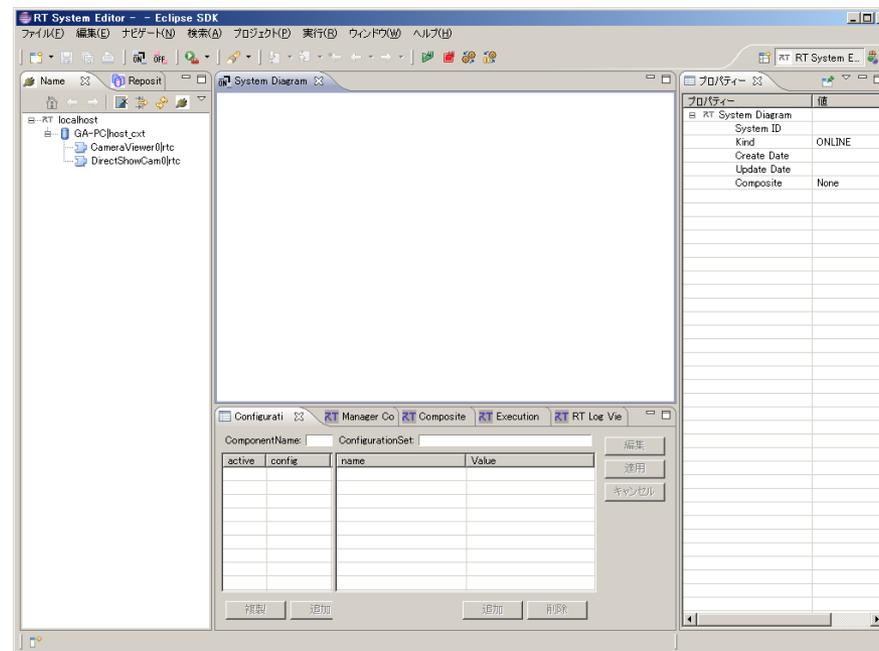
Eclipse上でツールの構成を管理する単位
メニュー、ツールバー、エディタ、ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

● ネームサービスへ接続

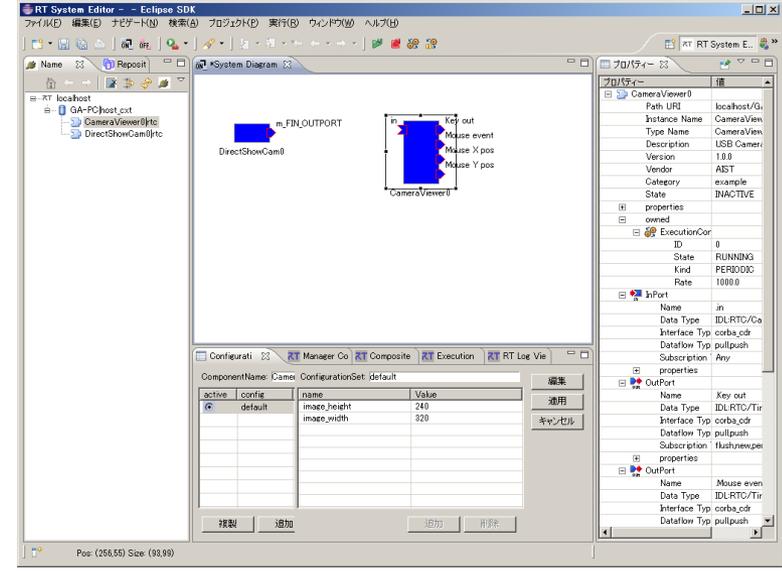
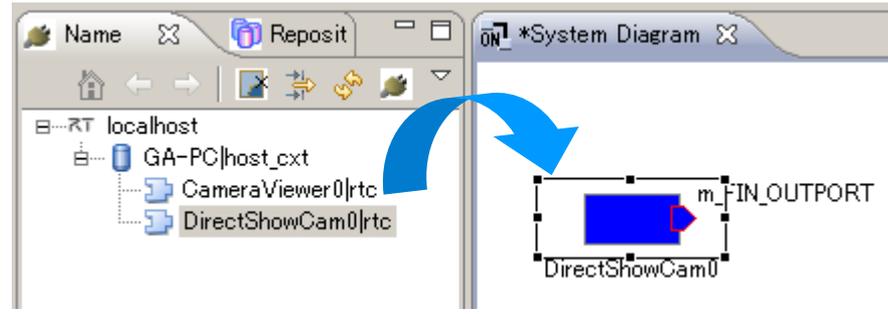


※対象ネームサーバのアドレス、ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能

● システムエディタの起動



● RTコンポーネントの配置



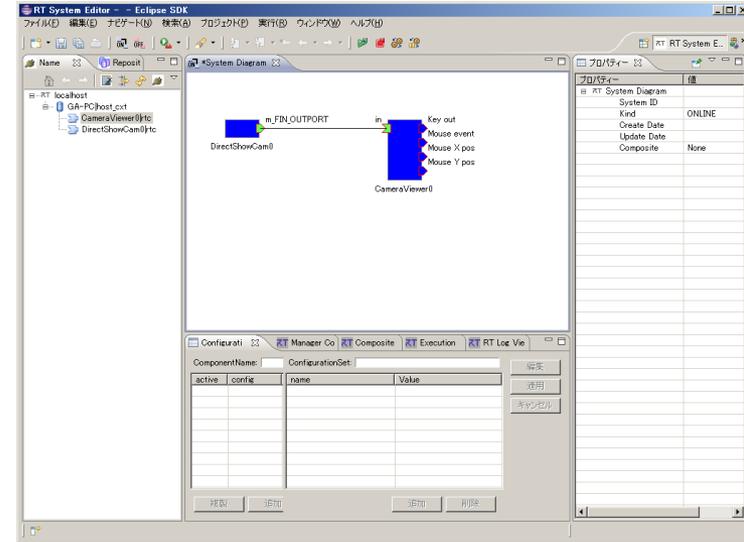
※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

● ポートの接続

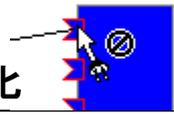
① 接続元のポートから接続先のポートまでドラッグ



② 【接続プロファイル】画面で「OK」を選択

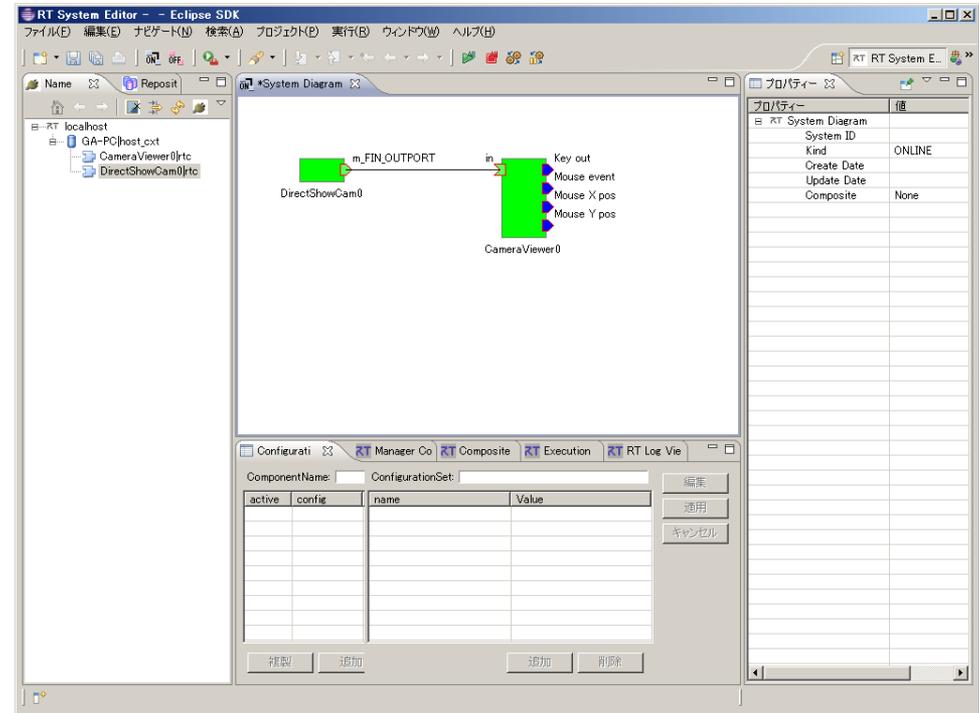
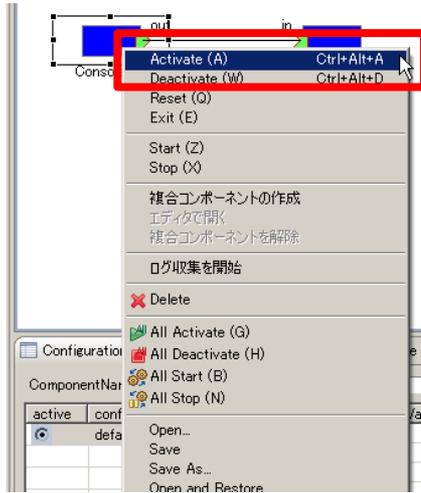


※ポートのプロパティが異なる場合など、接続不可能なポートの場合にはアイコンが変化

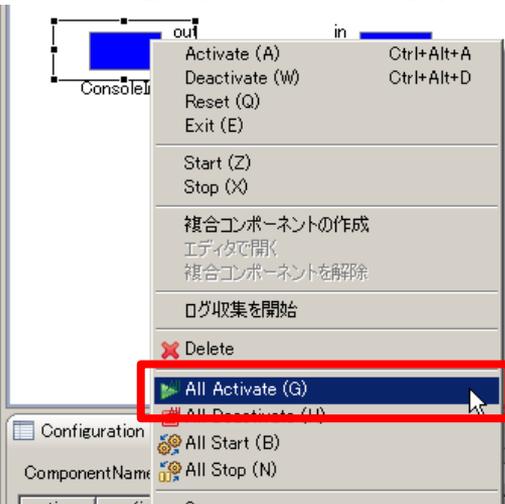


● コンポーネントの起動

※各RTC単位で起動する場合



※全てのRTCを一括で起動する場合



※停止はDeactivateを実行

※RTC間の接続を切る場合には接続線をDelete
もしくは、右クリックメニューから「Delete」を選択

| アクション名 | 説明 |
|------------|--|
| Activate | 対象RTCを活性化する |
| Deactivate | 対象RTCを非活性化する |
| Reset | 対象RTCをエラー状態からリセットする |
| Exit | 対象RTCの実行主体(ExecutionContext)を停止し, 終了する |
| Start | 実行主体(ExecutionContext)の動作を開始する |
| Stop | 実行主体(ExecutionContext)の動作を停止する |

■各コンポーネント単位での動作変更



■全コンポーネントの動作を一括変更



※ポップアップメニュー中でのキーバインドを追加

※単独RTCのActivate/Deactivateについては, グローバルはショートカットキー定義を追加

| 項目 | 設定内容 |
|------------------|--|
| Name | 接続の名称 |
| DataType | ポート間で送受信するデータの型. ex)TimedOctet, TimedShortなど |
| InterfaceType | データを送受信するポートの型. ex)corba_cdrなど |
| DataFlowType | データの送信方法. ex)push, pullなど |
| SubscriptionType | データ送信タイミング. 送信方法がPushの場合有効. New, Periodic, Flushから選択 |
| Push Rate | データ送信周期(単位:Hz). SubscriptionTypeがPeriodicの場合のみ有効 |
| Push Policy | データ送信ポリシー. SubscriptionTypeがNew, Periodicの場合のみ有効. all, fifo, skip, newから選択 |
| Skip Count | 送信データスキップ数. Push PolicyがSkipの場合のみ有効 |

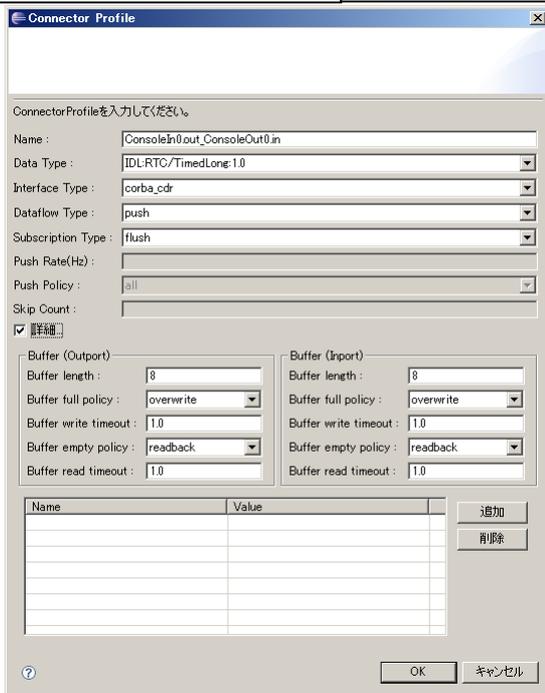
n SubscriptionType

- New : バッファ内に新規データが格納されたタイミングで送信
- Periodic : 一定周期で定期的にデータを送信
- Flush : バッファを介さず即座に同期的に送信

n Push Policy

- all : バッファ内のデータを一括送信
- fifo : バッファ内のデータをFIFOで1個ずつ送信
- skip : バッファ内のデータを間引いて送信
- new : バッファ内のデータの最新値を送信(古い値は捨てられる)

| 項目 | 設定内容 |
|----------------------|---|
| Buffer length | バッファの大きさ |
| Buffer full policy | データ書き込み時に、バッファフルだった場合の処理。 overwrite, do_nothing, blockから選択 |
| Buffer write timeout | データ書き込み時に、タイムアウトイベントを発生させるまでの時間(単位:秒) |
| Buffer empty policy | データ読み出し時に、バッファが空だった場合の処理。 readback, do_nothing, blockから選択 |
| Buffer read timeout | データ読み出し時に、タイムアウトイベントを発生させるまでの時間(単位:秒) |



- ※OutPort側のバッファ, InPort側のバッファそれぞれに設定可能
- ※timeoutとして「0.0」を設定した場合は、タイムアウトしない

● Buffer Policy

- overwrite : 上書き
- readback : 最後の要素を再読み出し
- block : ブロック
- do_nothing : なにもしない

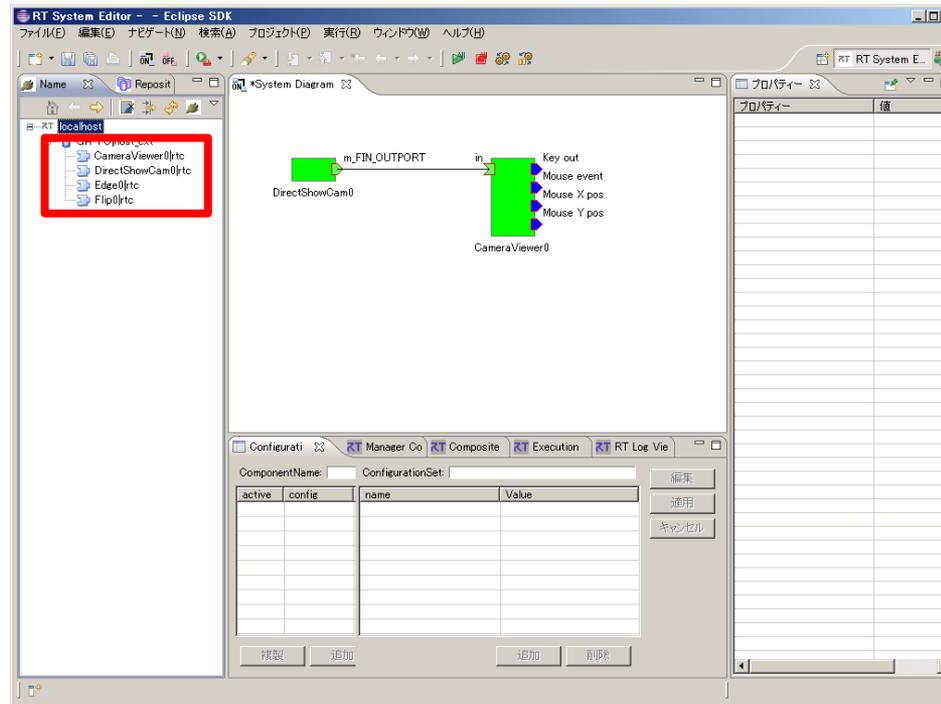
- ※Buffer Policy = Block+timeout時間の指定で、一定時間後読み出し/書き込み不可能な場合にタイムアウトを発生させる処理となる

- Flipコンポーネントの起動

[プログラム] → [OpenRTM-aist 1.1] → [C++] → [components]
 → [opencv-rtcs] → **[FlipComp.exe]**

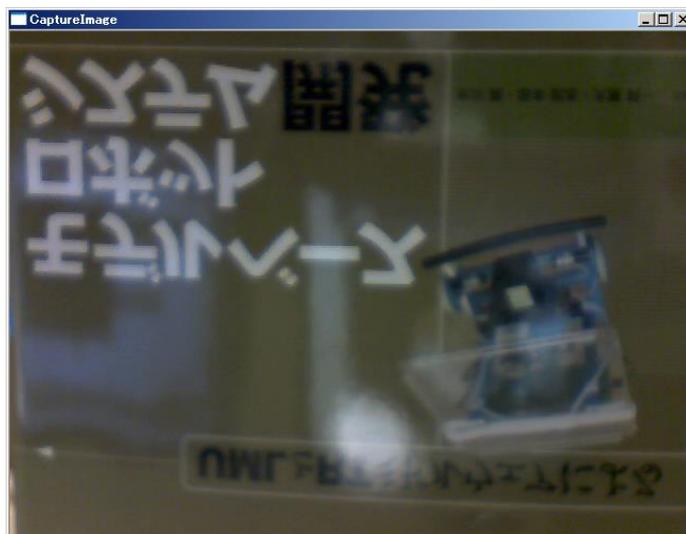
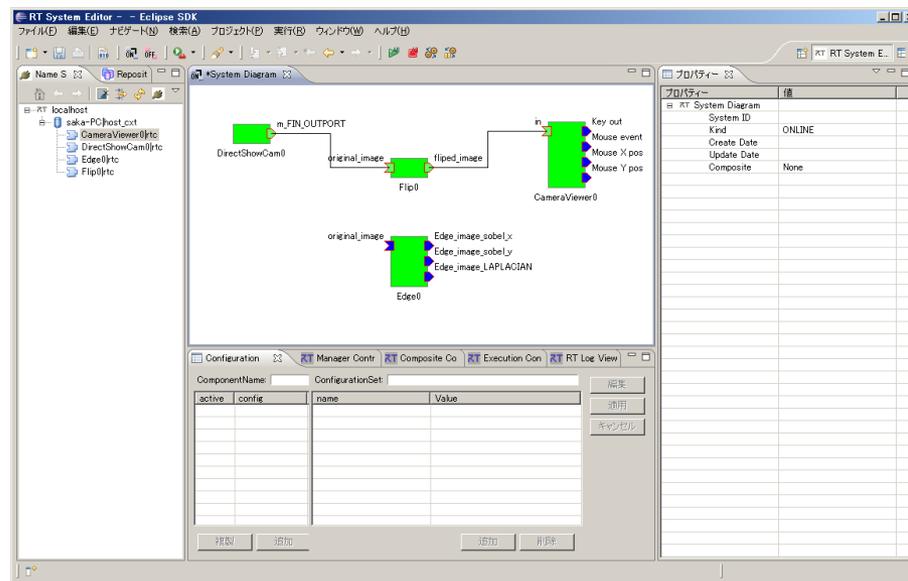
- Edgeコンポーネントの起動

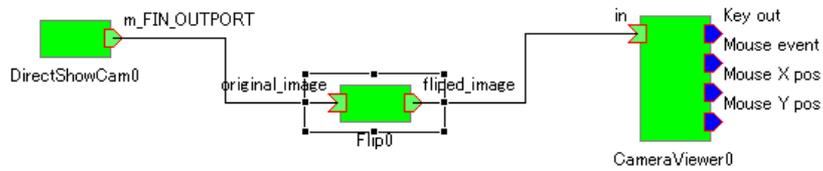
[プログラム] → [OpenRTM-aist 1.1] → [C++] → [components]
 → [opencv-rtcs] → **[EdgeComp.exe]**



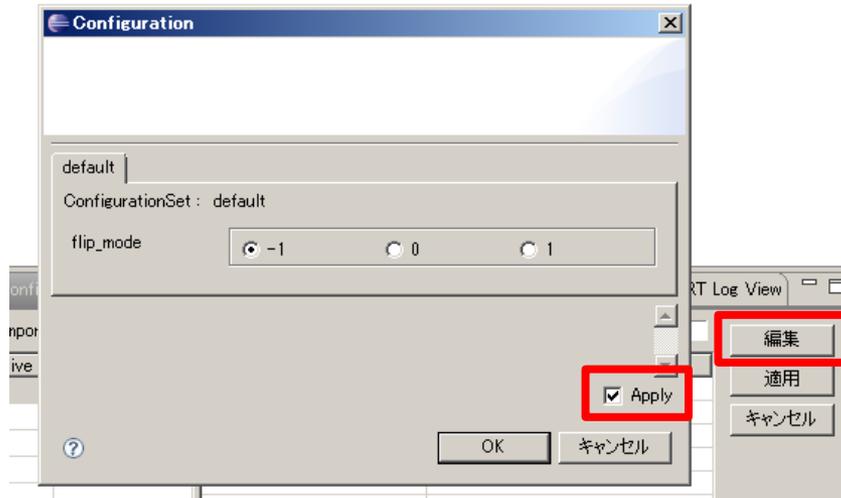
Flip側との接続

- DirectShowCam → Flip
 - CameraViewerと接続
(接続プロファイルはデフォルト設定)
- AllActivateを実行





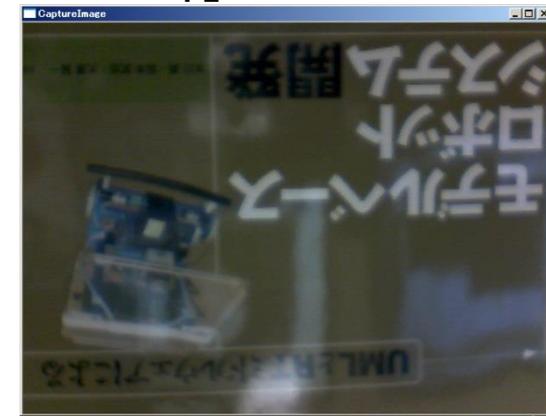
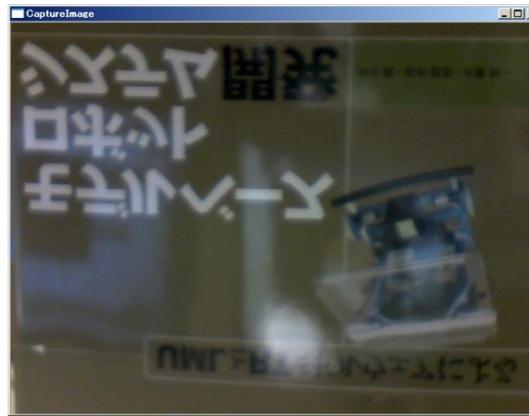
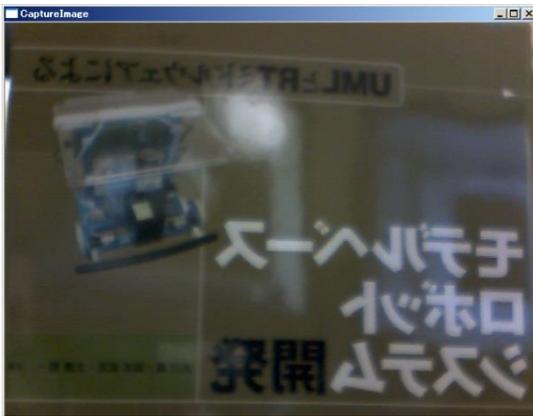
- ConfigurationViewの「編集」
- 表示されたダイアログ内で「flip_mode」の値を変更
- 「Apply」のチェックボックス



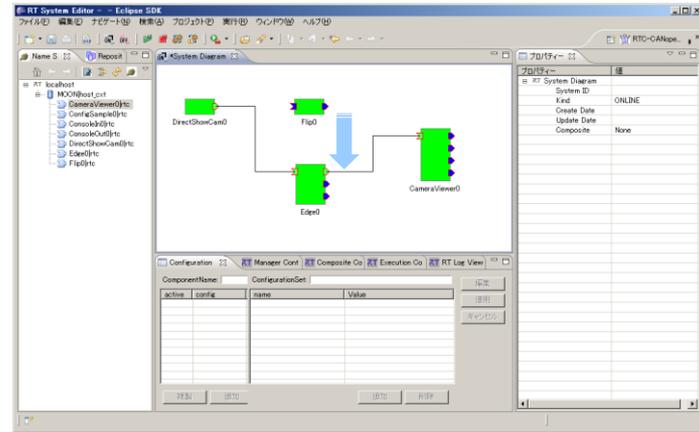
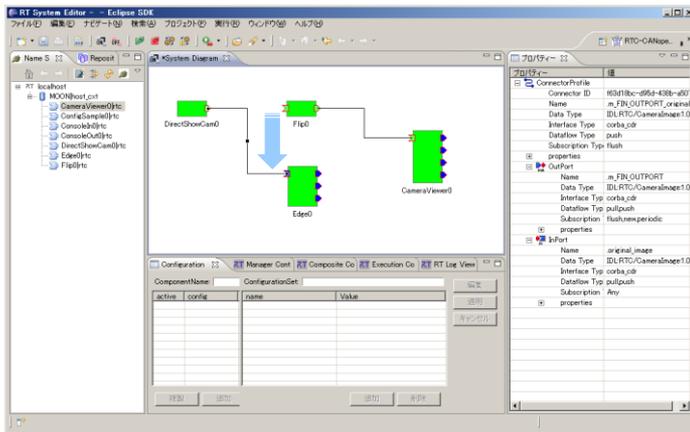
flip_mode=1

flip_mode=0

flip_mode=-1



- Edge側への差し替え
 - Flipに繋がっている接続線を選択
 - Flip側のPort部分に表示されているハンドルをEdge側のPortに繋ぎ替え
 - 接続プロファイルはデフォルト設定のまま



| コンポーネント名 | 処理内容 |
|-----------------------------|----------------------------|
| Affine | 入力画像をAffine変換 |
| BackGroundSubtractionSimple | 2枚の入力画像の差分抽出 |
| Binarization | 入力画像を2値化 |
| Chromakey | 入力画像から背景を取り除き, 別の背景に重ね合わせる |
| DilationErosion | 2値化後の画像に膨張縮小処理 |
| Edge | 入力画像のエッジ検出 |
| Findcontour | 入力画像の輪郭抽出処理 |
| Flip | 入力画像の反転処理 |
| Histogram | 入力画像内のヒストグラム算出処理 |
| Hough | 入力画像をハフ変換 |
| ImageSubstraction | 差分画像の抽出 |
| ObjectTracking | 指定物体を画像内から抽出 |
| Perspective | 遠近透過処理 |
| RockPaperScissors | 手の形状認識, グー/チョキ/パーの認識 |
| Rotate | 指定角度での画像回転 |
| Scale | 入力画像の指定倍への拡大/縮小処理 |
| SepiaComp | 入力画像のセピア化処理 |

- 全て[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]→[opencv-rtcs]内のサンプル

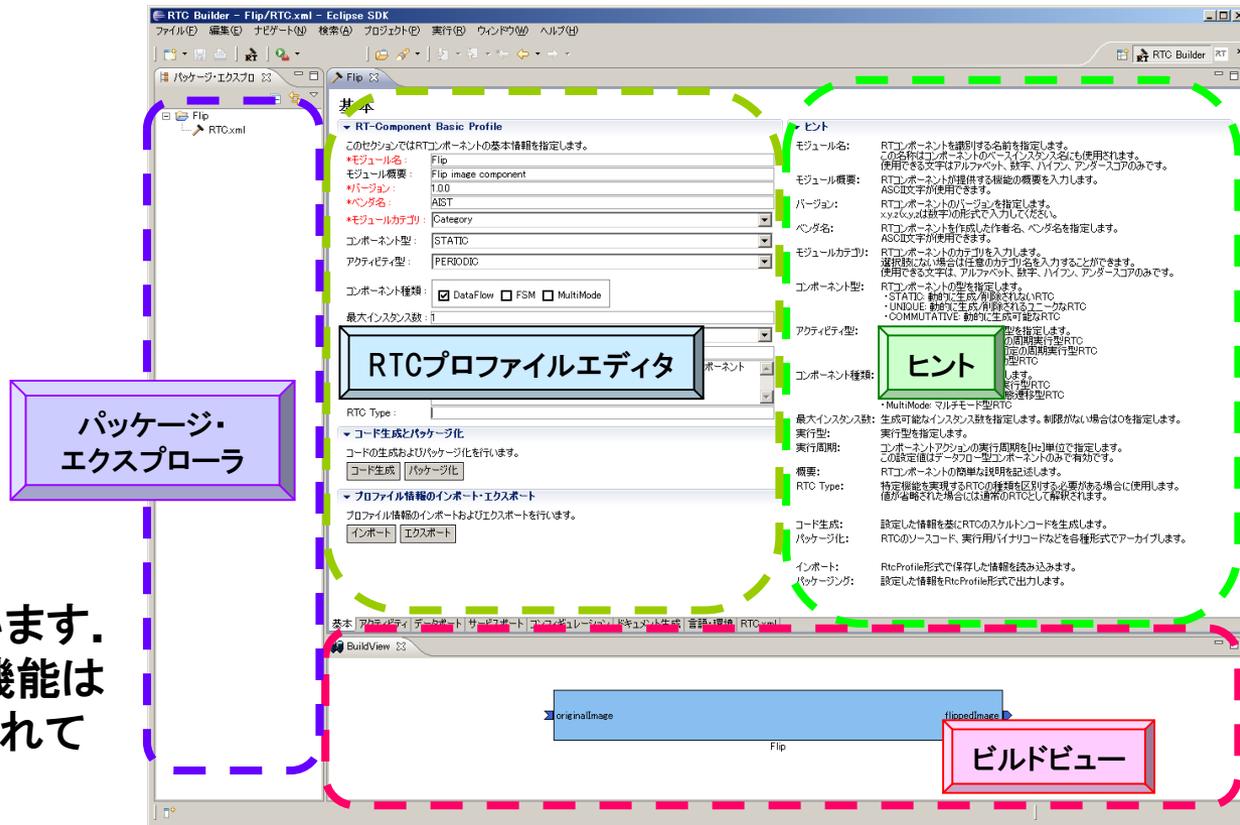
コンポーネント開発ツール RTCBuilderについて

■ RTCBuilderとは？

- コンポーネントのプロファイル情報を入力し、ソースコード等の雛形を生成するツール
- 開発言語用プラグインを追加することにより、各言語向けRTCの雛形を生成することが可能

- C++
- Java
- Python

- ※C++用コード生成機能はRtcBuilder本体に含まれています。
- ※その他の言語用コード生成機能は追加プラグインとして提供されています



① ツールバー内のアイコンをクリック



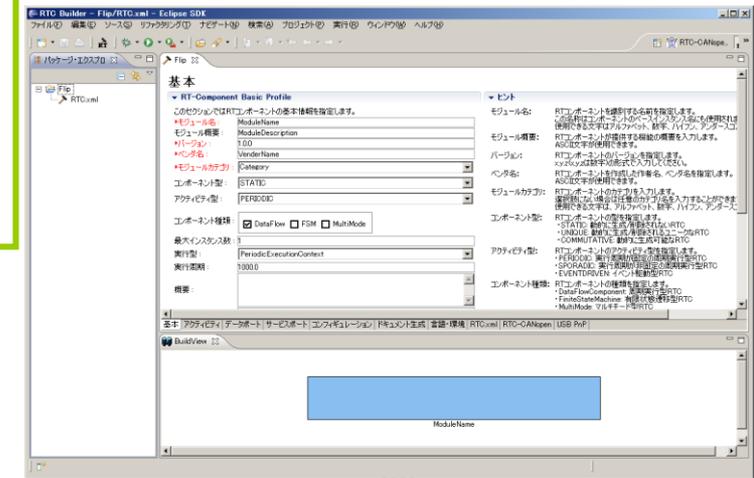
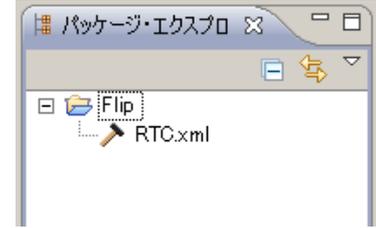
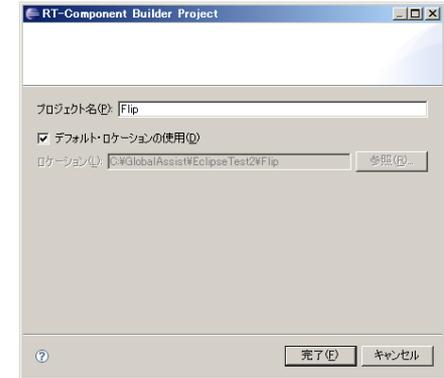
- ※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し、「次へ」
- ※メニューから「ファイル」-「Open New Builder Editor」を選択

※任意の場所にプロジェクトを作成したい場合

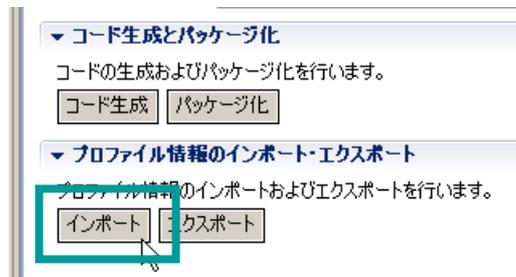
- ②にて「デフォルト・ロケーションの使用」チェックボックスを外す
- 「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名: Flip

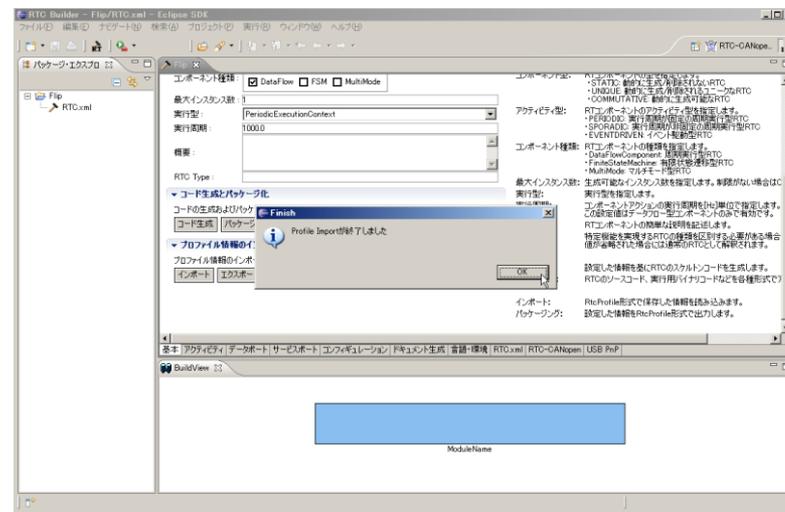
② 「プロジェクト名」欄に入力し、「終了」



①「基本」タブ下部の「インポート」ボタンをクリック



②【インポート】画面にて対象ファイルを選択



- 作成済みのRTコンポーネント情報を再利用
 - 「エクスポート」機能を利用して出力したファイルの読み込みが可能
 - コード生成時に作成されるRtcProfileの情報を読み込み可能
 - XML形式, YAML形式での入出力が可能

● コード生成

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**

▼ **プロフィール情報のインポート・エクスポート**

プロフィール情報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 **エクスポート**

Information dialog box: Generate success. OK

File Explorer: Flip

- cmake_modules
 - cmake_uninstall.cmake
 - CPackWDXcmake
- cpack_resources
 - Description.txt
 - License.rtf
 - License.txt
 - wix.xsl.in
- CMakeLists.txt
- Doxyfile.in
- Flip.conf
- Flip.cpp
- Flip.h
- FlipComp.cpp
- rtc.conf
- RTC.xml
- RTC.xml20111116204517

● コード生成実行後、パースペクティブを自動切替

Dialog box: 関連付けられたパースペクティブを開きますか? このプロジェクトは C/C++ パースペクティブに関連付けられます。このパースペクティブを開きますか? はい いいえ

MultiMode: マルチモード型 RIG
 最大インスタンス数: 生成可能なインスタンス数を指定し実行型を指定します。
 実行型: コンポーネントアクションの実行周期
 実行周期: この設定値はデフォルト型コンポーネントの簡単な説明を表現する RTC の種類が異なる場合、通常の RIG の情報、生成したプロジェクトは以下の方法で WorkSpace 内に直接入力

コード生成: 設定した情報を基に RTC のスケルトン
 パッケージ化: RTC のソースコード、実行用バイナリ

※生成コードが表示されない場合には、「リフレッシュ」を実行

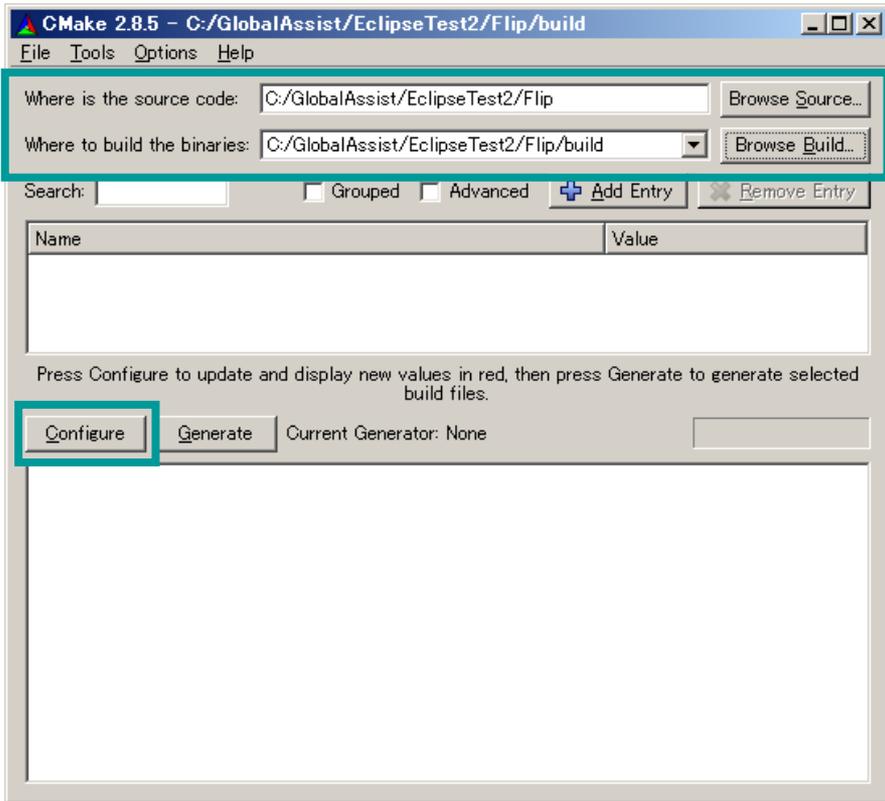
C++版RTC → CDT

Java版RTC → JDT

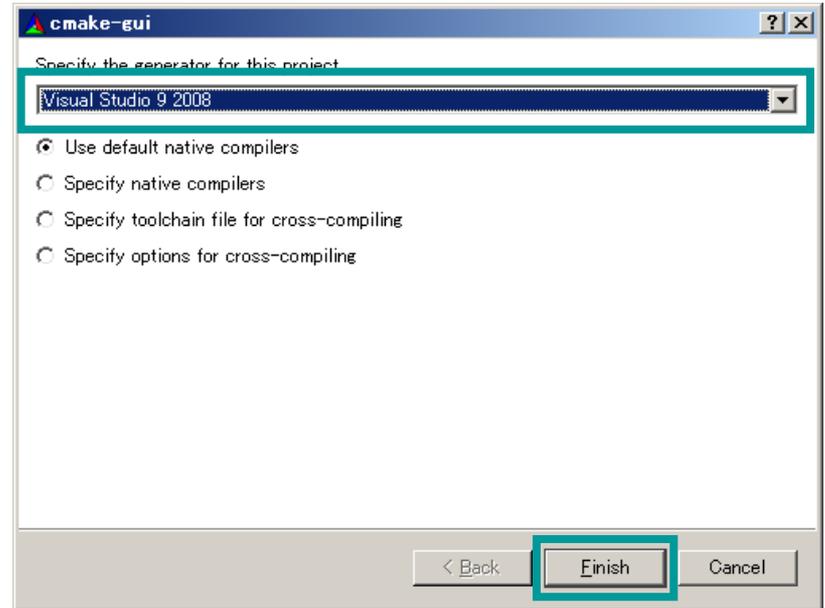
(デフォルトインストール済み)

Python版 → PyDev

① GUI版Cmakeを起動し, source, binaryのディレクトリを指定



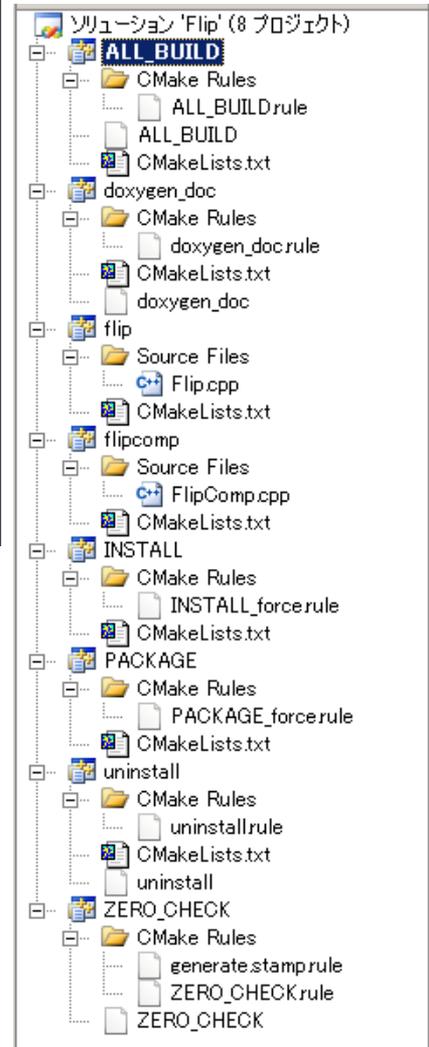
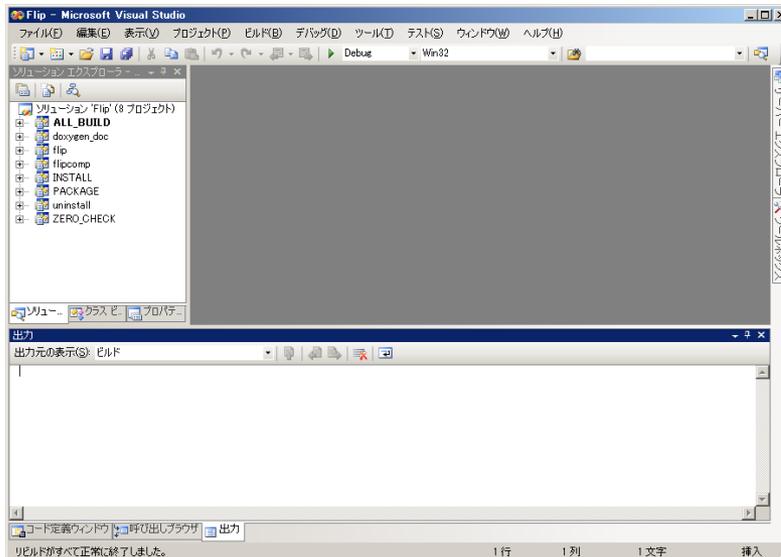
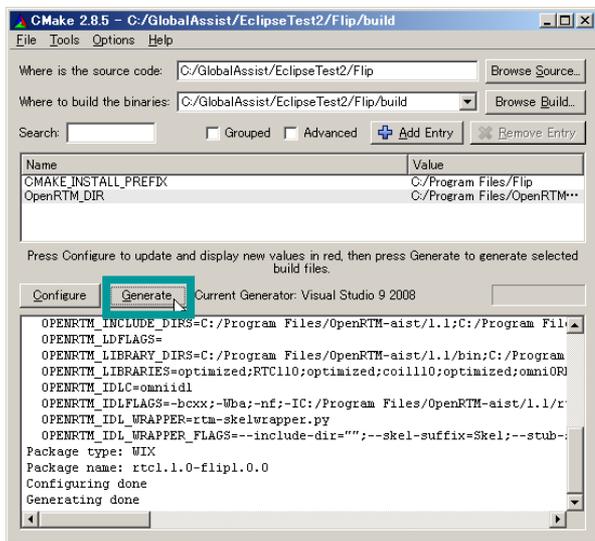
② 「Configure」を実行し, 使用するプラットフォームを選択



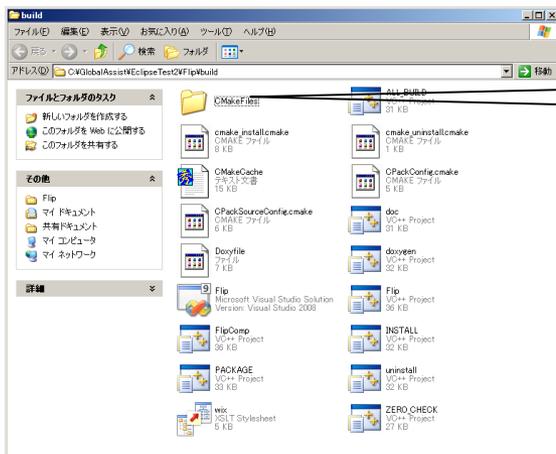
※binaryには, sourceとは別のディレクトリを指定する事を推奨

※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

③ 正常終了後、「Generate」を実行



④ binaryとして指定したディレクトリ内にあるソリューションファイルを開き、「ソリューションをビルド」を実行



| 画面要素名 | 説明 |
|----------------|--|
| 基本プロフィール | RTコンポーネントのプロファイル情報など、コンポーネントの基本情報を設定。 コード生成, インポート/エクスポート, パッケージング処理を実行 |
| アクティビティ・プロフィール | RTコンポーネントがサポートしているアクティビティ情報を設定 |
| データポート・プロフィール | RTコンポーネントに付属するデータポートに関する情報を設定 |
| サービスポート・プロフィール | RTコンポーネントに付属するサービスポートおよび各サービスポートに付属するサービスインターフェースに関する情報を設定 |
| コンフィギュレーション | RTコンポーネントに設定するユーザ定義のコンフィギュレーション・パラメータセット情報およびシステムのコンフィギュレーション情報を設定 |
| ドキュメント生成 | 生成したコードに追加する各種ドキュメント情報を設定 |
| 言語・環境 | 生成対象コードの選択やOSなどの実行環境に関する情報を設定 |
| RTC.xml | 設定した情報を基に生成したRTC仕様(RtcProfile)を表示 |

● RTコンポーネントの名称など、基本的な情報を設定

基本

▼ RT-Component Basic Profile

このセクションではRTコンポーネントの基本情報を指定します。

***モジュール名:** Flip

モジュール概要: Flip image component

***バージョン:** 1.0.0

***ベンダ名:** AIST

***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: DataFlow FSM MultiMode

最大インスタンス数: 1

実行型: PeriodicExecutionContext

実行周期: 0.0

概要: OpenCVライブラリのうち、cvFlip関数を用いて画像の反転を行うコンポーネント

RTC Type:

▼ ヒント

モジュール名: RTコンポーネントを識別する名前を指定します。この名称はコンポーネントのベースインスタンス名にも使用されます。使用できる文字はアルファベット、数字、ハイフン、アンダースコアのみです。

モジュール概要: RTコンポーネントが提供する機能の概要を入力します。ASCII文字が使用できます。

モジュール名: Flip

モジュール概要: 任意(Flip image component)

バージョン: 1.0.0

ベンダ名: 任意(AIST)

モジュールカテゴリ: 任意(Category)

コンポーネント型: STATIC

アクティビティ型: PERIODIC

コンポーネントの種類: DataFlow

最大インスタンス数: 1

実行型: PeriodicExecutionContext

実行周期: 1000.0

▼ 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。

▼ プロファイル情報のインポート・エクスポート

プロファイル情報のインポートおよびエクスポートを行います。

- ※エディタ内の項目名が赤字の要素は必須入力項目
- ※画面右側は各入力項目に関する説明

● 生成対象RTCで実装予定のアクティビティを設定

アクティビティ

このセクションでは使用するアクションコールバックを指定します。

コンポーネントの初期化と終了処理に関するアクション

onInitialize onFinalize

実行コンテキストの起動と停止に関するアクション

onStartup onShutdown

alive状態でのコンポーネントアクション

onActivated onDeactivated onAborting

onError onReset

Dataflow型コンポーネントのアクション

onExecute onStateUpdate onRateChanged

FSM型コンポーネントのアクション

onAction

Mode型コンポーネントのアクション

onModeChanged

Documentation

このセクションでは各アクションの概要を説明するドキュメントを記述します。上段のアクションを選択すると、それぞれのドキュメントを記述できます。

アクティビティ名: onInitialize ON OFF

動作概要: コンポーネント自身の各種初期化処理

事前条件: なし

事後条件: コンポーネントの初期化処理が正常に完了している

ヒント

onInitialize: 初期化処理です。コンポーネントライフサイクル開始時に一度だけ呼びれます。常に有効。

onFinalize: 終了処理です。コンポーネントライフサイクルの終了時に1度だけ呼びれます。

onStartup: ExecutionContextが実行を開始するとき1度だけ呼びれます。

onShutdown: ExecutionContextが実行を停止するとき1度だけ呼びれます。

onActivated: 非アクティブ状態からアクティブ化される時1度だけ呼びれます。

onDeactivated: アクティブ状態から非アクティブ化される時1度だけ呼びれます。

onAborting: ERROR状態に入る前に1度だけ呼びれます。

onError: ERROR状態にある間周期的に呼びれます。

onReset: ERROR状態からリセットされ非アクティブ状態に移行するとき1度だけ呼びれます。

onExecute: アクティブ状態時に周期的に呼びれます。

onStateUpdate: onExecuteの後毎回呼びれます。

onRateChanged: ExecutionContextのrateが変更されたとき呼びれます。

onAction: 対応する状態に応じた動作を実行するために呼びれます。

onModeChanged: モードが変更された時に呼びれます。

動作概要: アクティビティの概要説明を記述します。

事前条件: アクティビティを実行する前に成立すべき事前条件を記述します。

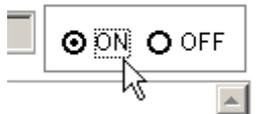
事後条件: アクティビティを実行した後に成立すべき事後条件を記述します。

基本 | アクティビティ | データポート | サービスポート | コンフィギュレーション | ドキュメント生成 | 言語・環境 | RTC.xml | Mapping ID | USB PnP | RTC-CANopen

① 設定対象のアクティビティを選択



② 使用/未使用を設定



以下をチェック:
onActivated
onDeactivated
onExecute

- ※現在選択中のアクティビティは、一覧画面にて赤字で表示
- ※使用(ON)が選択されているアクティビティは、一覧画面にて背景を水色で表示
- ※各アクティビティには、「動作概要」「事前条件」「事後条件」を記述可能
 →記述した各種コメントは、生成コード内にDoxygen形式で追加される

● 生成対象RTCに付加するDataPortの情報を設定

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 |
|----------------|--------|-----------------|--------|
| originalImage | | flippedImage | |
| | Delete | | Delete |

ヒント

データポート: RTCコンポーネント間でデータを出力するOutPortとInPortを接続する。

InPort: RTCコンポーネントに他のRTCコンポーネント(OutPort)の情報を設定します。

OutPort: RTCコンポーネントから他のRTCコンポーネントにデータを出力する。

ポート名: データポートを識別するポート名は、同一のコンポーネントに対して一意なASCII文字が使用できます。

データ型: データポート間でやり取りするデータの型は、OpenRTMで使用することが出来ます。

変数名: データポートに関連付いた変数の名称は言語に依存します。

ポートの場所: RTSystemEditorなどのこのプロファイルはオプションで指定することが出来ます。

ドキュメント: データポートに関する情報を記述する必要がある場合は、このドキュメントに記述する必要があります。

Detail

このセクションではデータポート毎の概要を説明するドキュメントを記述します。上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名: originalImage (InPort)

*データ型: RTC::CameraImage

変数名: originalImage

表示位置: LEFT

Documentation: キャプチャされた画像データ

概要説明:

データ型: CameraImage型(OpenRTM-aistのInterfaceDataTypes.idlにて定義されているデータ型)

データ数: 任意

意味: 反転処理の対象となる画像データ

単位: なし

① 該当種類の欄の「Add」ボタンをクリックし、ポートを追加後、直接入力で名称設定

ポートの情報を設定します。

*ポート名 (OutPort)

dp_name

Add

Delete

② 設定する型情報を一覧から選択

Detail

このセクションではデータポート毎の概要を説明するドキュメントを記述します。上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名: originalImage (InPort)

*データ型: RTC::CameraImage

変数名: RTC::BumperArrayGeometry

表示位置: RTC::CameraImage

Documentation:

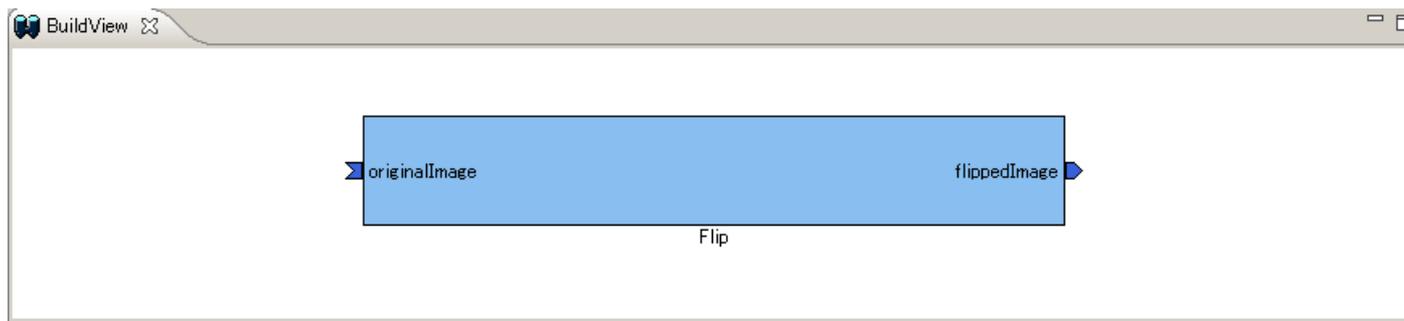
※ データ型は、型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

※ OpenRTM-aistにて事前定義されている型については、デフォルトで使用可能
→ [RTM_Root]rtm/idl 以下に存在するIDLファイルで定義された型

※ 各ポートに対する説明記述を設定可能

→ 記述した各種コメントは、生成コード内にDoxygen形式で追加される

※Portの設定内容に応じて、下部のBuildViewの表示が変化



● InPort

ポート名: originalImage

データ型: RTC::CameraImage

変数名: originalImage

表示位置: left

● OutPort

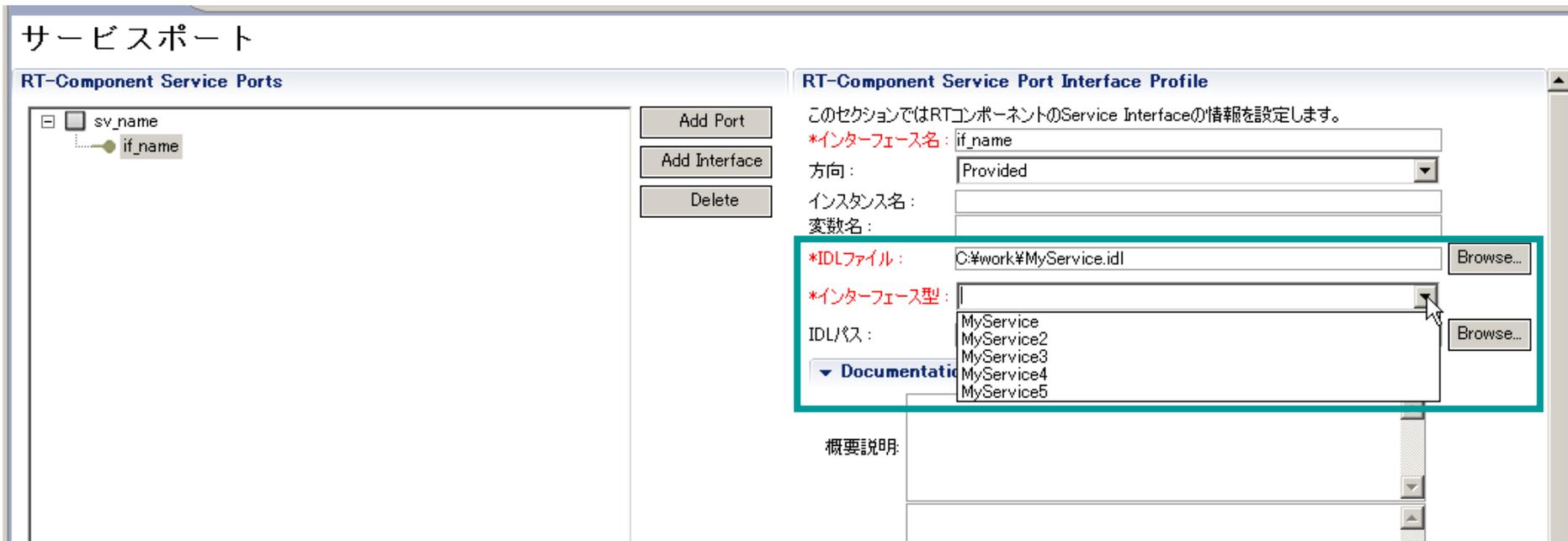
ポート名: flippedImage

データ型: RTC::CameraImage

変数名: flippedImage

表示位置: right

- 生成対象RTCに付加するServicePortの情報を設定



- サービスインターフェースの指定
 - IDLファイルを指定すると, 定義されたインターフェース情報を表示

今回のサンプルでは未使用

n 生成対象RTCで使用する設定情報を設定

コンフィギュレーション・パラメータ

▼ RT-Component Configuration Parameter Definitions
このセクションではRTコンポーネントのコンフィギュレーション・パラメータを指定します。

| *名称 | |
|----------|--|
| flipMode | |
| | |
| | |
| | |

Add Delete

▼ Detail
このセクションでは各コンフィギュレーション・パラメータの詳細情報を指定します。

パラメータ名: flipMode

*データ型: int

*デフォルト値: 0

変数名: flipMode

単位:

制約条件: (-1,0,1)

Widget: radio

Step:

Documentation

データ名: flipMode

デフォルト値: 0

概要説明: 画像の反転方法を指定するパラメータ

単位: なし

データ範囲: -1,0,1

制約条件: 0: 上下反転したい場合
1: 左右反転したい場合
-1: 上下左右反転したい場合

▼ ヒント

Config. Param: RTコンポーネントの再初期化パラメータ

パラメータ名: コンフィギュレーションパラメータ名

データ型: コンフィギュレーション基本型

デフォルト値: コンフィギュレーションRTコンポーネントの解釈

変数名: コンフィギュレーション実体

単位: コンフィギュレーション

制約条件: コンフィギュレーション指定範囲(100以内の整数・列挙配列・ハッシュ)

Widget: コンフィギュレーション

Step: 設定

①「Add」ボタンをクリックし、追加後、直接入力で名称設定

▼ RT-Component Configuration Parameter Definitions
このセクションではRTコンポーネントのコンフィギュレーション・パラメータを指定します。

| *名称 | |
|------------|--|
| conf_name0 | |
| | |
| | |
| | |

Add Delete

②詳細画面にて、型情報、変数名などを設定

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

- ※データ型は、short,int,long,float,double,stringから選択可能(直接入力も可能)
- ※制約情報とWidget情報を入力することで、RTSystemEditorのコンフィギュレーションビューの表示を設定することが可能

- 制約条件について

- データポートとコンフィギュレーションに設定可能
- チェックはあくまでもコンポーネント開発者側の責務
 - ミドルウェア側で検証を行っているわけではない

- 制約の記述書式

- 指定なし: 空白
- 即値: 値そのもの
 - 例) 100
- 範囲: $<$, $>$, $<=$, $>=$
 - 例) $0 \leq x \leq 100$
- 列挙型: (値1, 値2, ...)
 - 例) (val0, val1, val2)
- 配列型: 値1, 値2, ...
 - 例) val0, val1, val2
- ハッシュ型: { key0: 値0, key1: 値1, ... }
 - 例) { key0: val0, key1: val1 }

- Widget

- text(テキストボックス)
 - デフォルト
- slider(スライダ)
 - 数値型に対して範囲指定の場合
 - 刻み幅をstepにて指定可能
- spin(スピナ)
 - 数値型に対して範囲指定の場合
 - 刻み幅をstepにて指定可能
- radio(ラジオボタン)
 - 制約が列挙型の場合に指定可能

※ 指定したWidgetと制約条件がマッチしない場合は、テキストボックスを使用

● 生成対象RTCを実装する言語，動作環境に関する情報を設定

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++
- Python
- Java
- Ruby

Use old build environment.

ヒント

言語: RTコンポーネントを作成する言語を選択します。リスト中の言語から選択可能です。

環境: 言語ごとのライブラリの依存関係や、使用するOSなどの環境を選択します。

詳細情報で設定した内容(OS情報、ライブラリ情報など)は、プロファイル内にもみ保存されます。

環境

このセクションでは依存するライブラリや使用するOSなどを指定します

| Version | OS | |
|---------|----|--------|
| | | Add |
| | | Delete |
| | | |
| | | |
| | | |
| | | |

詳細情報

| OS Version | |
|------------|--------|
| | Add |
| | Delete |

| CPU | |
|-----|--------|
| | Add |
| | Delete |

このチェックボックスをONにすると、旧バージョンと同様なコード(Cmake を利用しない形式)を生成

「C++」を選択

Flipコンポーネントの実装

- CMakeで生成したソリューションファイルをVisual C++で開く
- Flip.cppの実装
 - onInitialize: コンポーネントの初期化
 - onActivate: データ保存領域の確保
 - onDeactivate: データ保存領域の解放
 - onExecute: Flip処理

