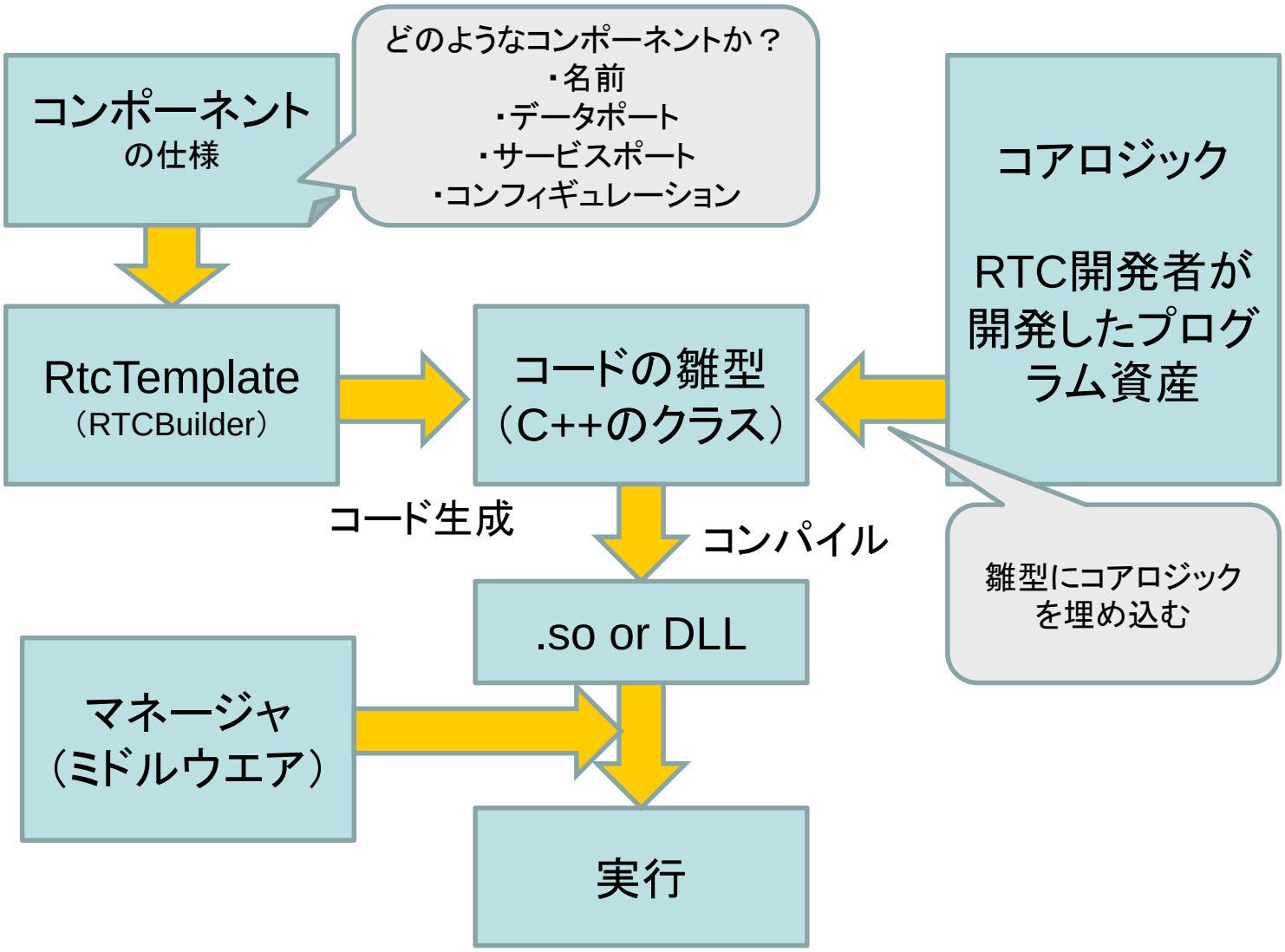
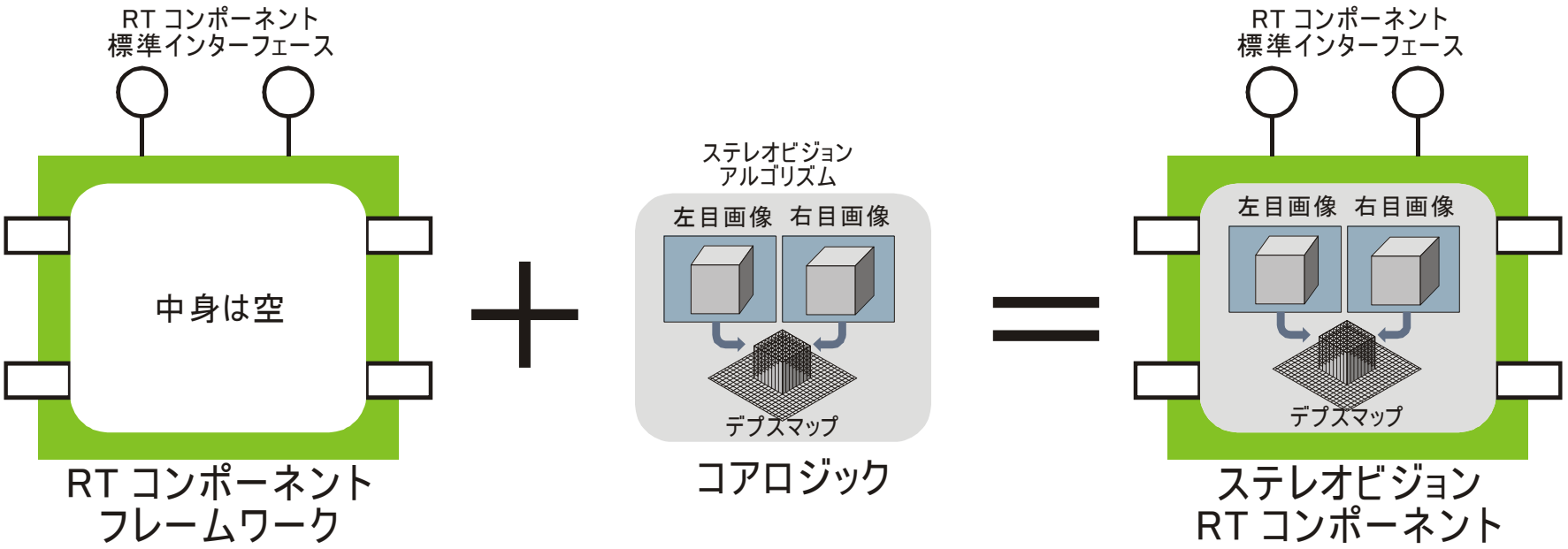
第2部 RTコンポーネントの開発

OpenRTMを使った開発の流れ

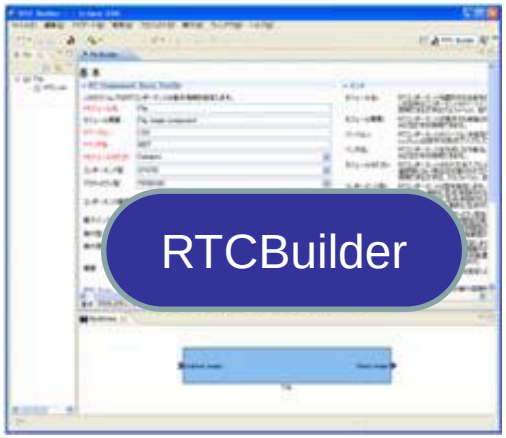


フレームワークとコアロジック

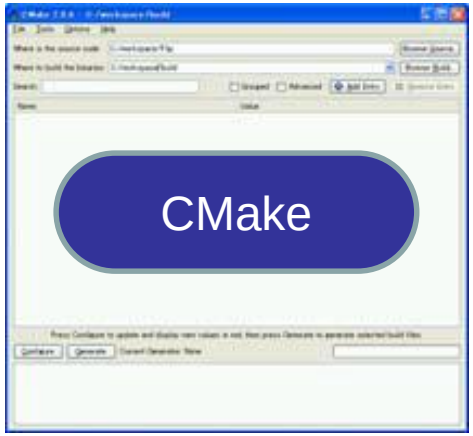


RTCフレームワーク＋コアロジック＝RTコンポーネント

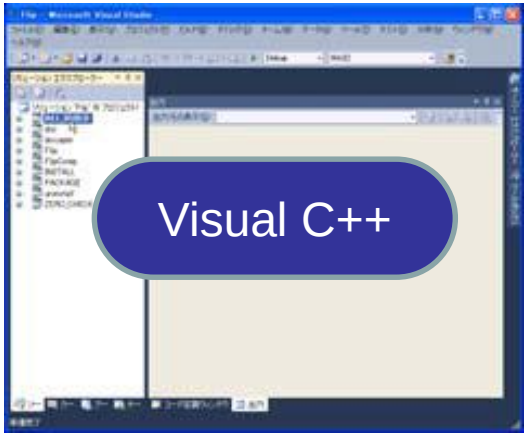
コンポーネントの作成
（Windowsの場合）



コンポーネントの
仕様の入力



VCのプロジェクト
ファイルの生成

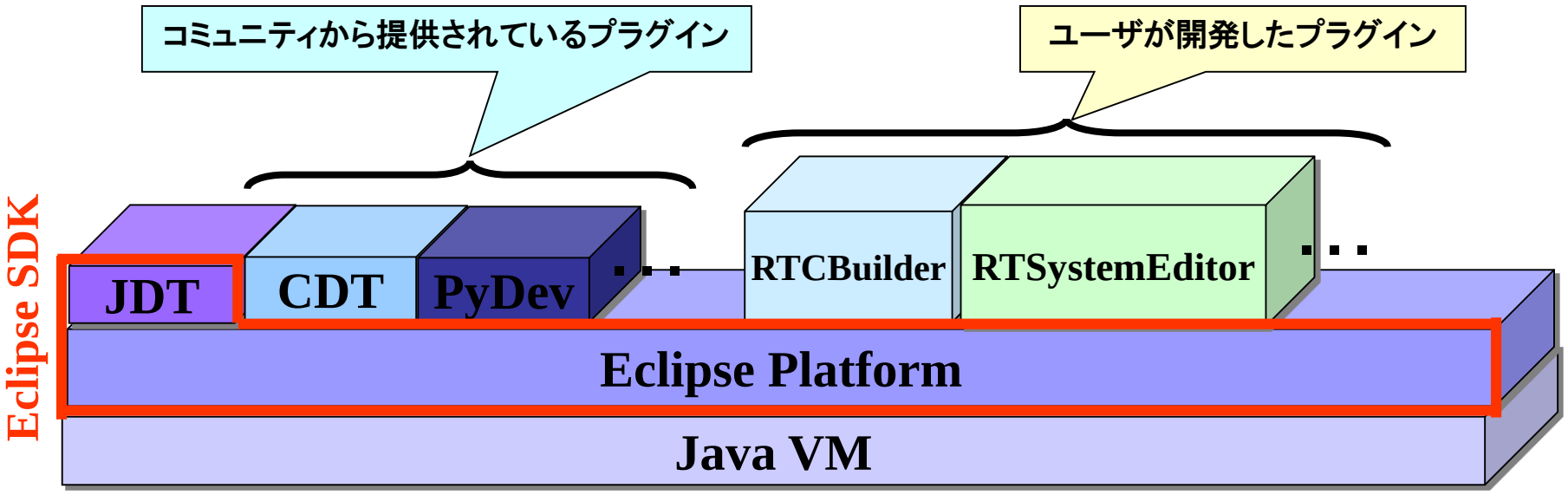


実装および
VCでコンパイル
実行ファイルの生成

テンプレートコード
の生成

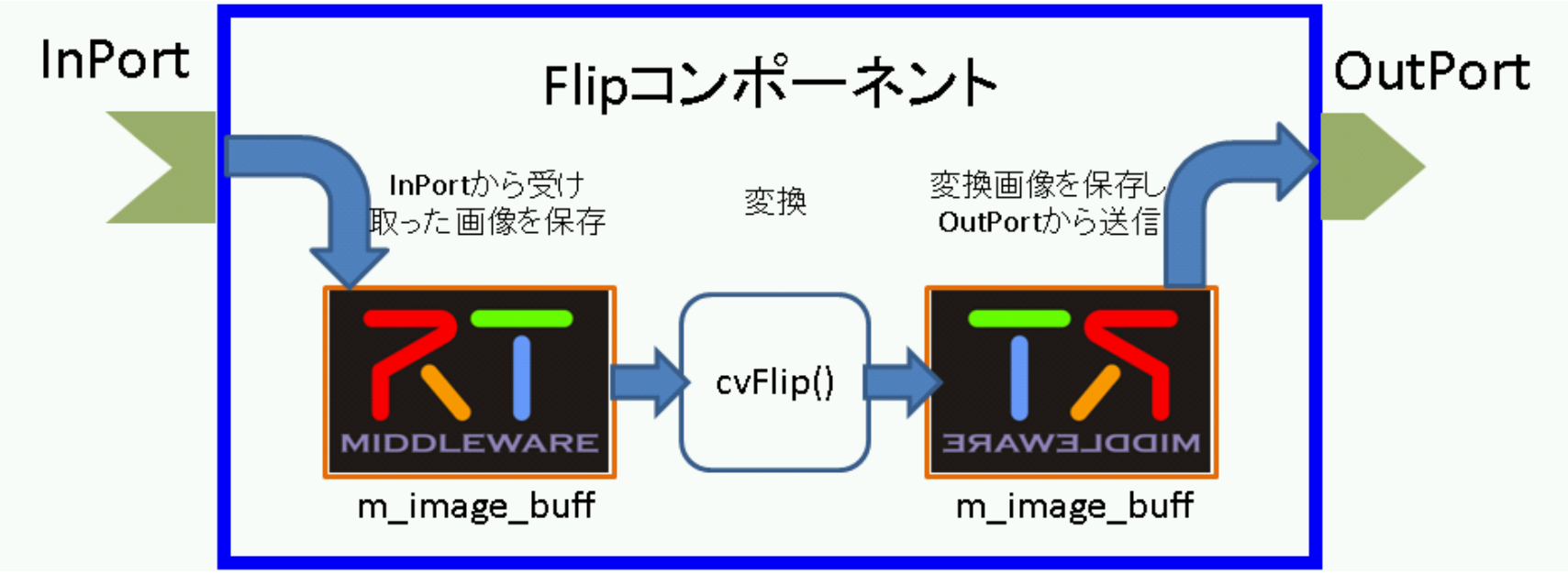
統合開発環境Eclipse

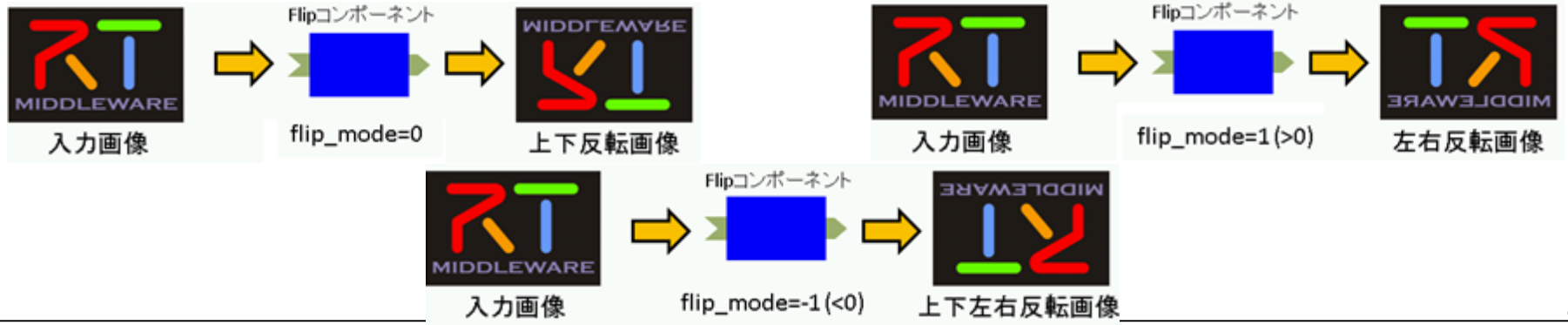
- オープンソース・コミュニティで開発されている統合開発環境
  - マルチプラットフォーム対応. WindowsやLinuxなど複数OS上で利用可能
  - 「Plug-in」形式を採用しており，新たなツールの追加，機能のカスタマイズが可能
  - RCP(Rich Client Platform)を利用することで，簡単に単独アプリ化が可能



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

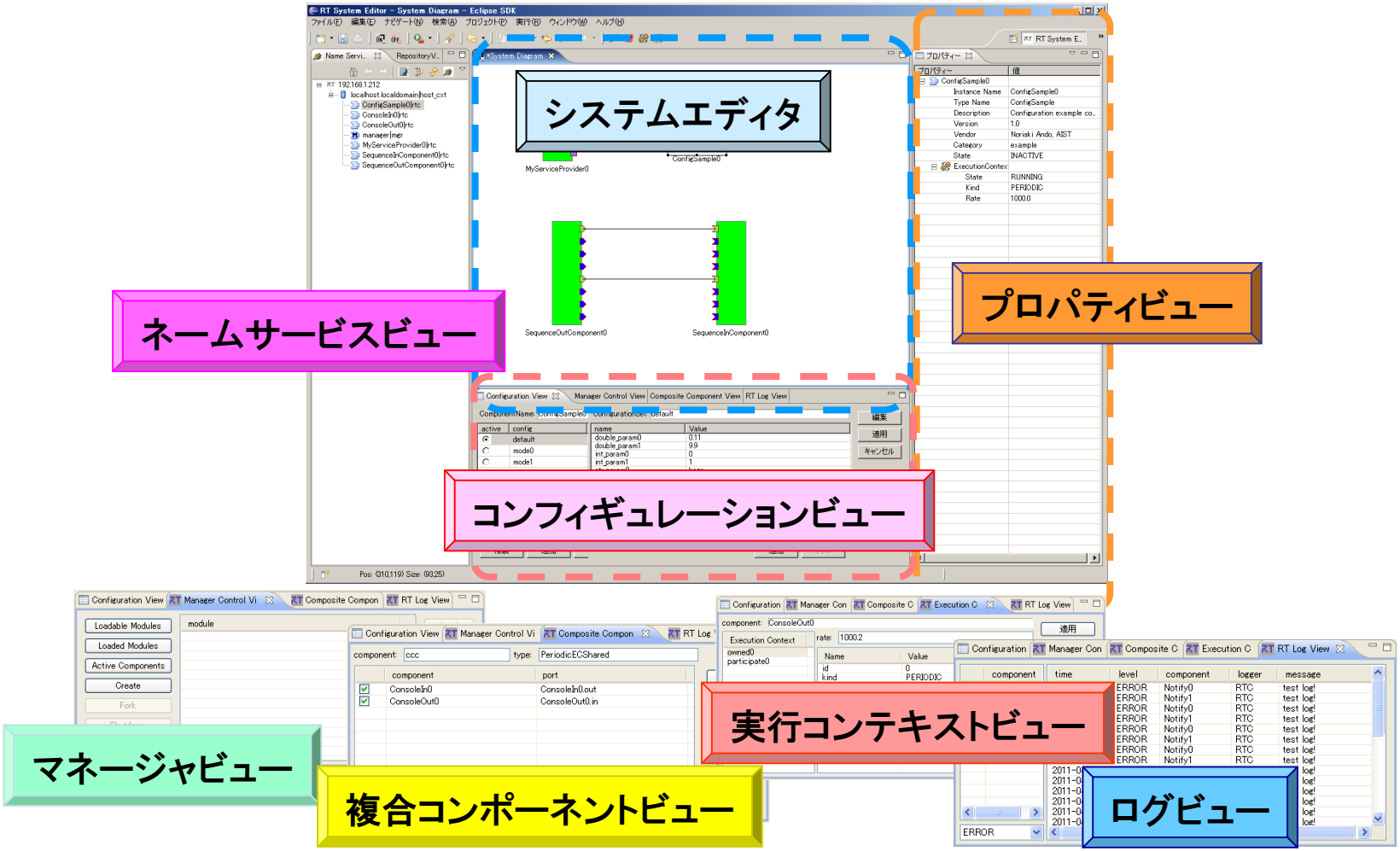




システム構築支援ツール
RTSystemEditorについて

RTSystemEditor概要

- 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



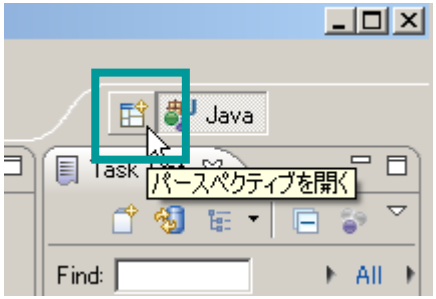
カメラ関連コンポーネントの起動

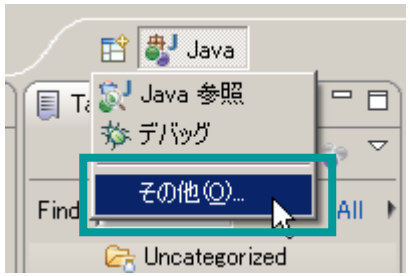
- Naming Serviceの起動
  - [スタート]メニューから
  - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]

- 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]

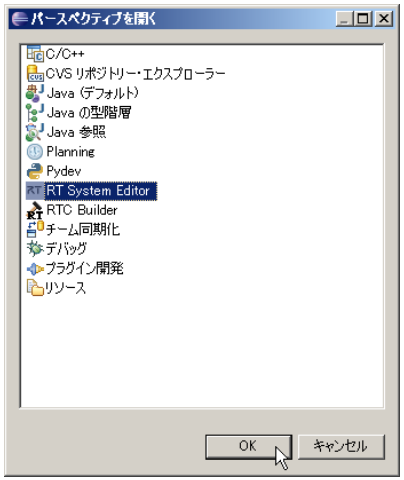
パースペクティブの切り替え

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



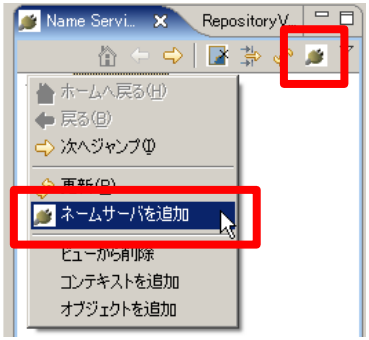


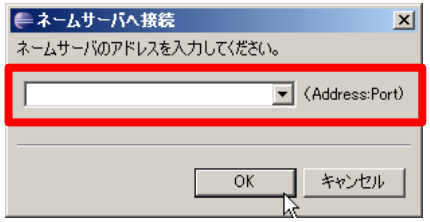
②一覧画面から対象ツールを選択

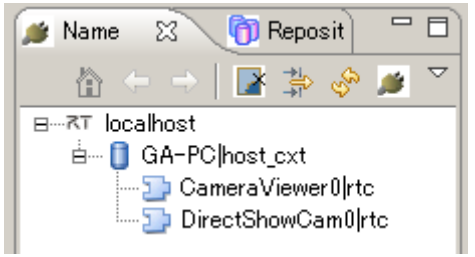


※パースペクティブ
Eclipse上でツールの構成を管理する単位
メニュー，ツールバー，エディタ，ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

- ネームサービスへ接続

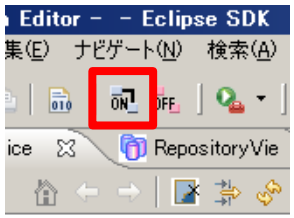


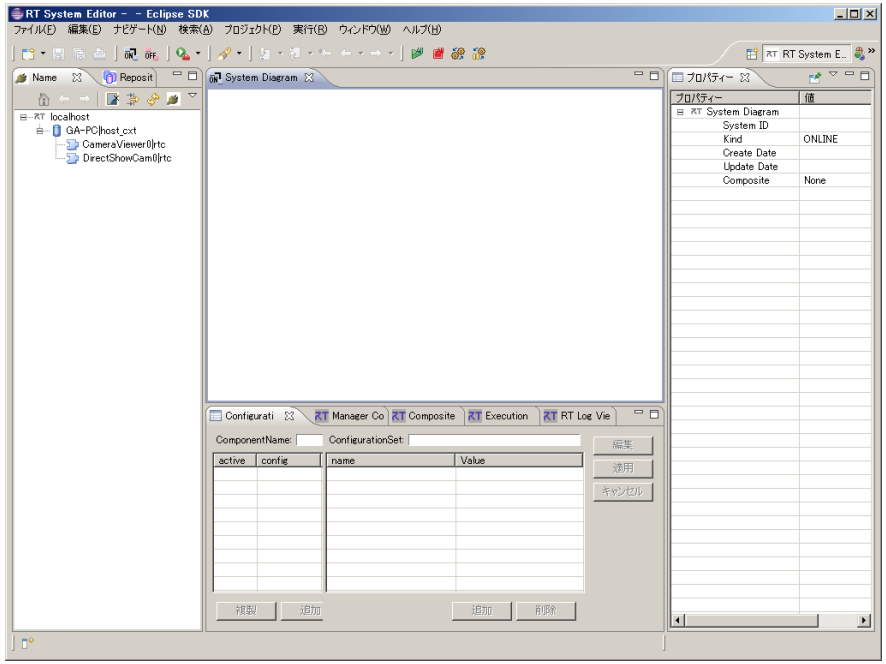




※対象ネームサーバのアドレス，ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能

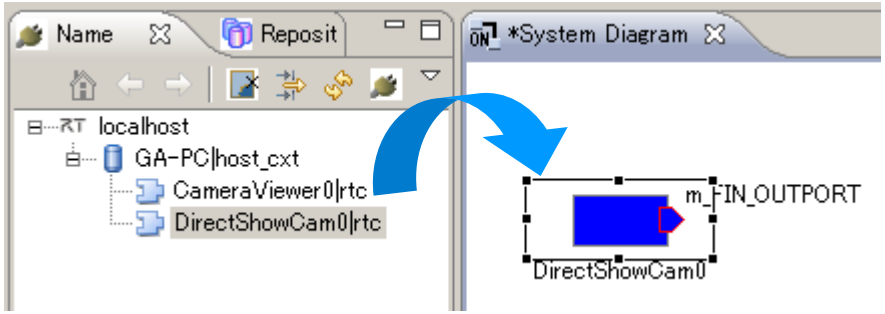
- システムエディタの起動

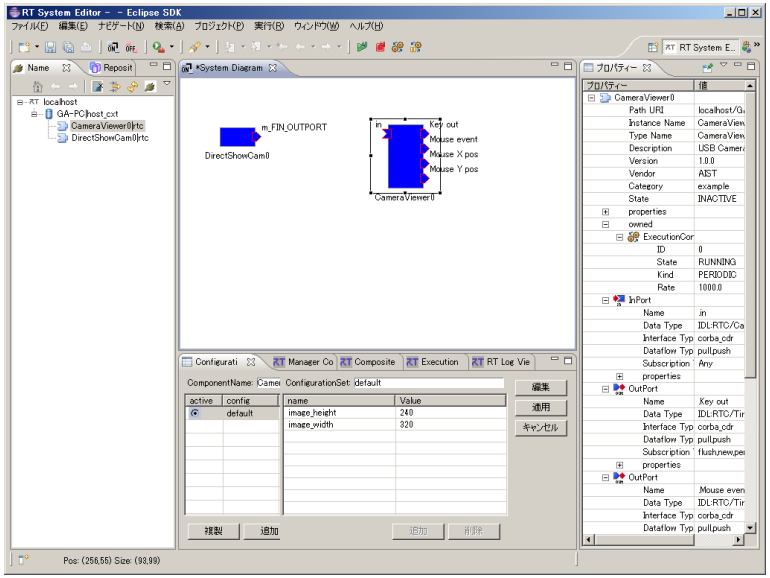




RTシステム構築の基本操作

- RTコンポーネントの配置

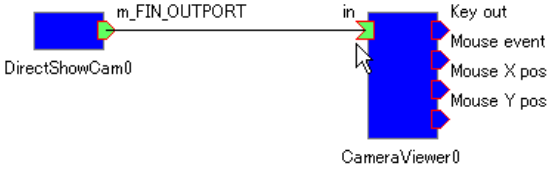




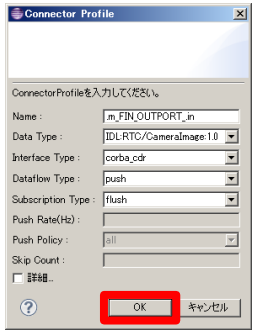
※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

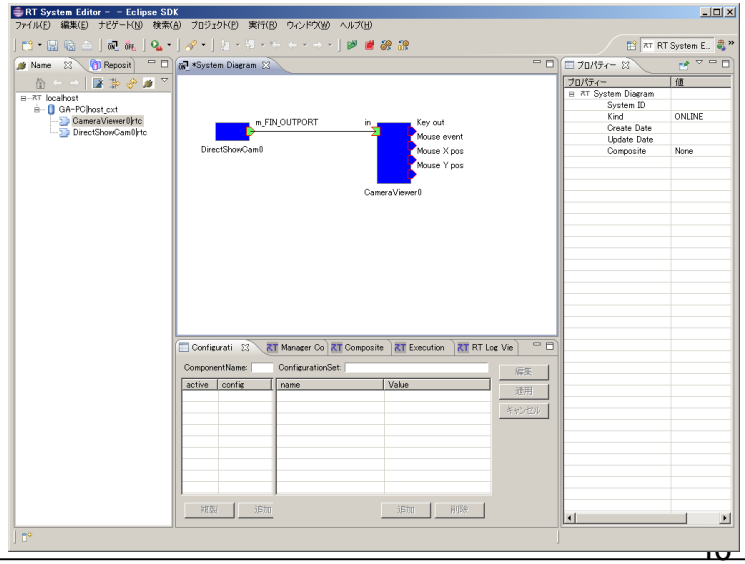
- ポートの接続

①接続元のポートから接続先のポートまでドラッグ

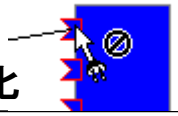


②【接続プロファイル】画面で「OK」を選択



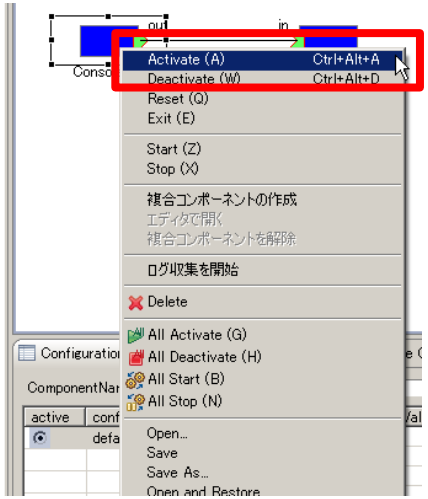


※ポートのプロパティが異なる場合など，接続不可能なポートの場合にはアイコンが変化

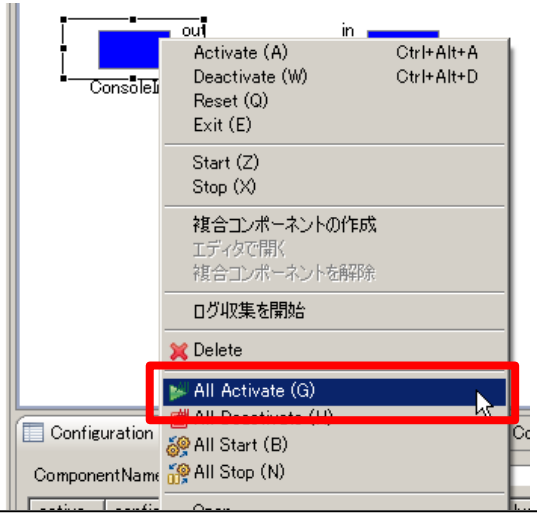


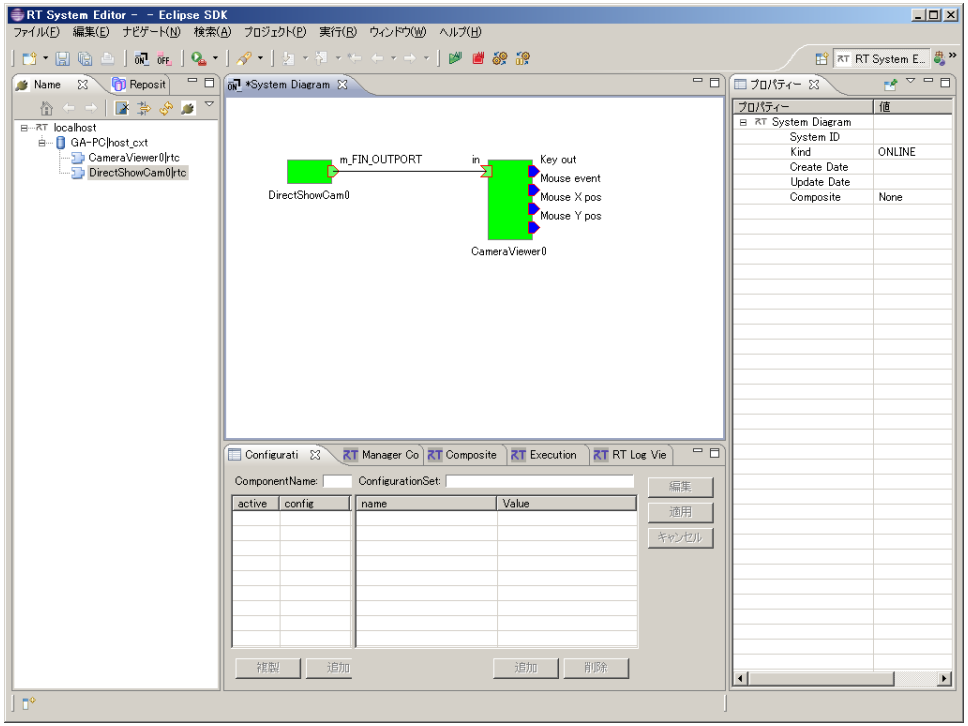
- コンポーネントの起動

※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

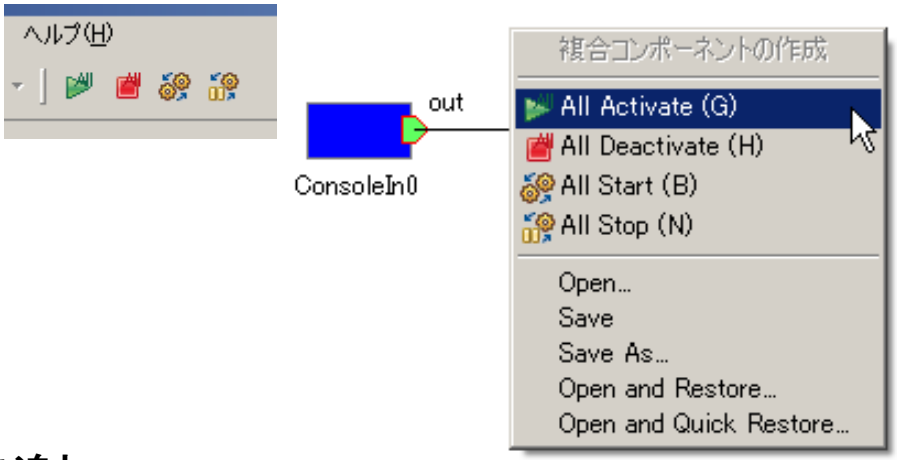
※RTC間の接続を切る場合には接続線をDelete
もしくは，右クリックメニューから「Delete」を選択

RTコンポーネントの動作

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

接続プロファイル(DataPort)について

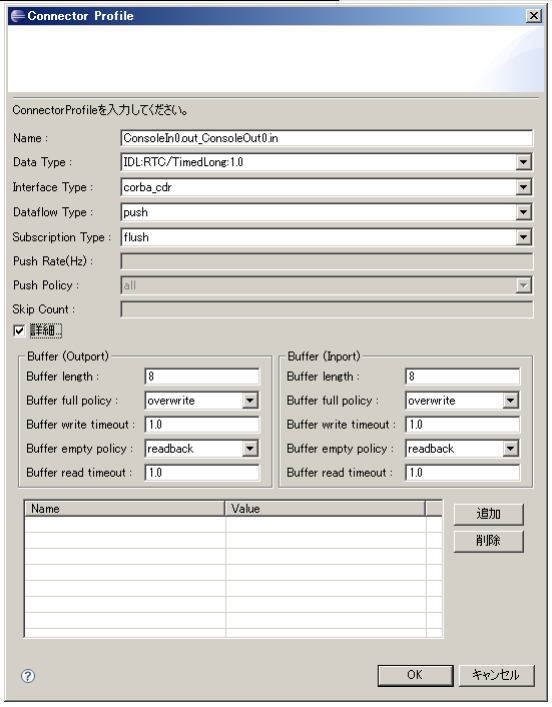
項目	設定内容
Name	接続の名称
DataType	ポート間で送受信するデータの型. ex)TimedOctet,TimedShortなど
InterfaceType	データを送受信するポートの型. ex)corba_cdrなど
DataFlowType	データの送信方法. ex)push, pullなど
SubscriptionType	データ送信タイミング. 送信方法がPushの場合有効. New, Periodic, Flushから選択
Push Rate	データ送信周期(単位:Hz). SubscriptionTypeがPeriodicの場合のみ有効
Push Policy	データ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

- 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

- 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)

接続プロファイル(DataPort)について

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

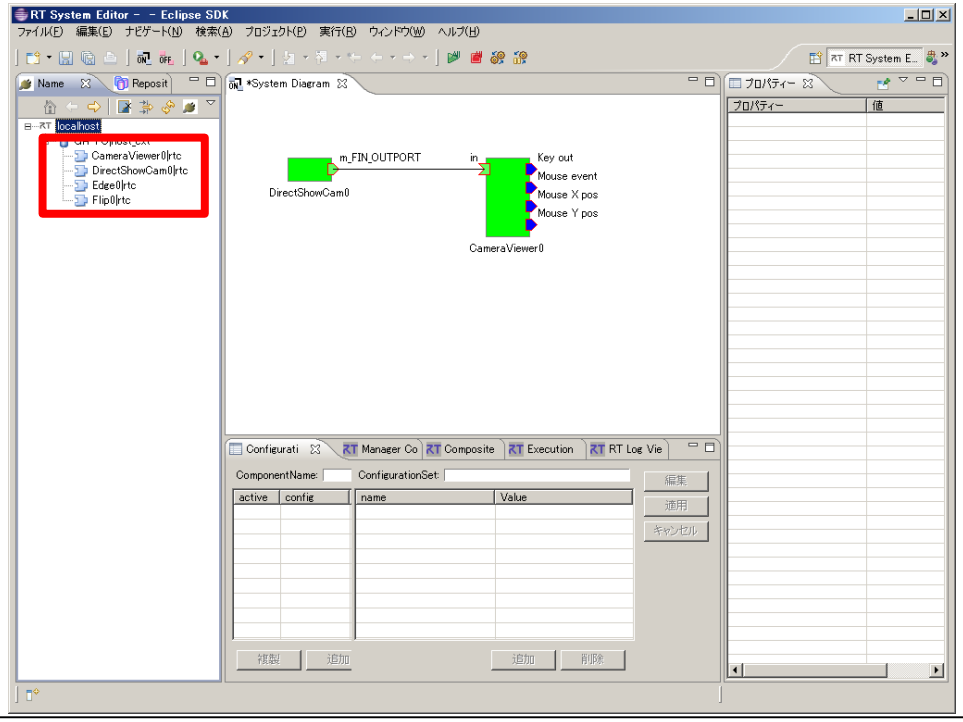
画像処理関連コンポーネントの起動

- 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]

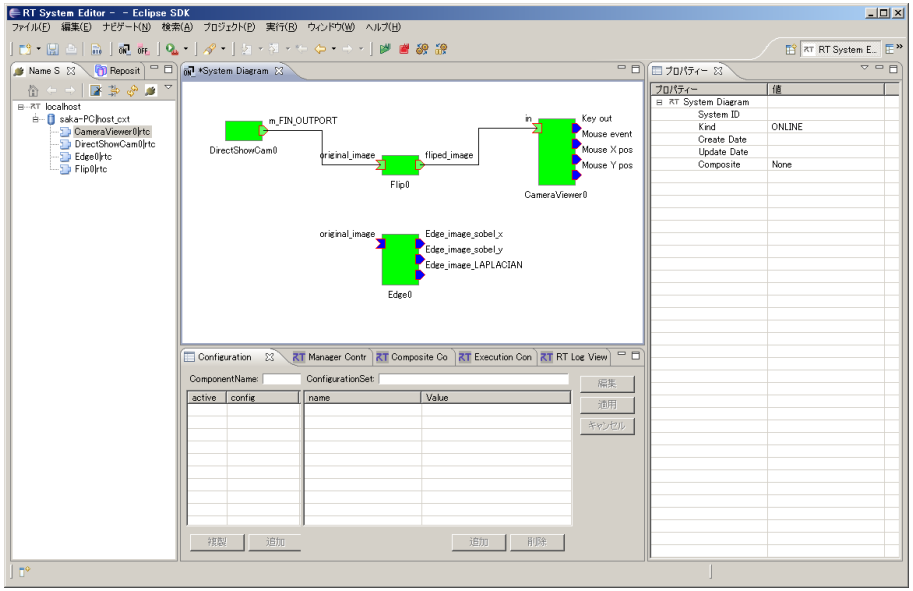
- Edgeコンポーネントの起動

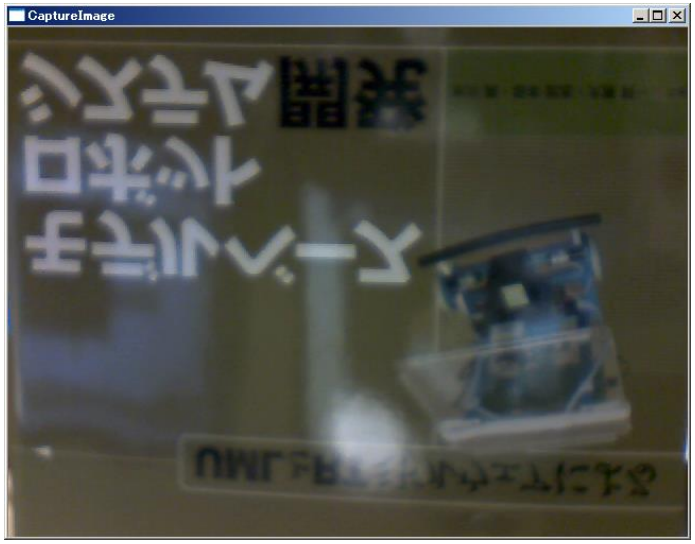
  [プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
  →[opencv-rtcs]→ [EdgeComp.exe]



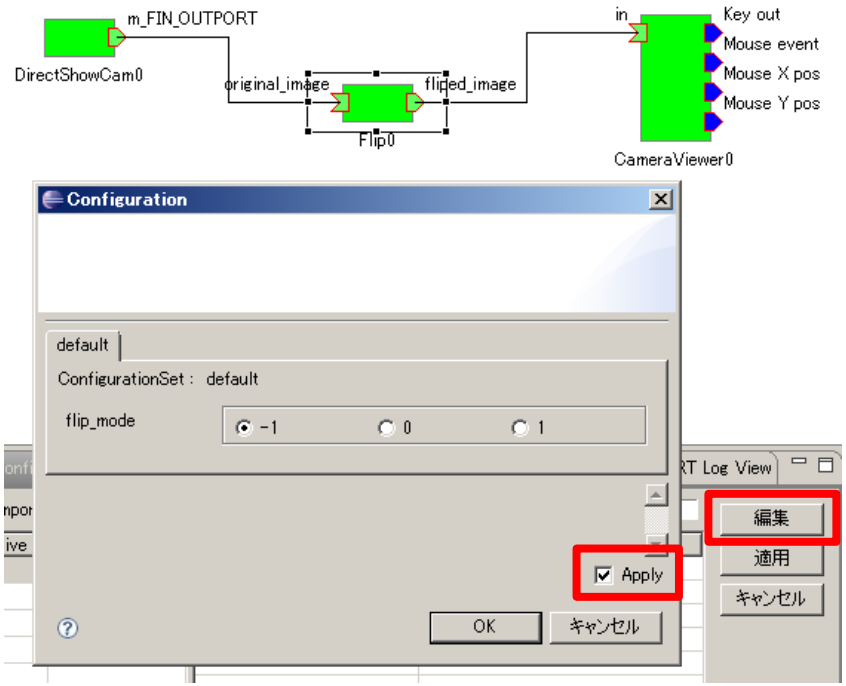
システムの構成

- 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



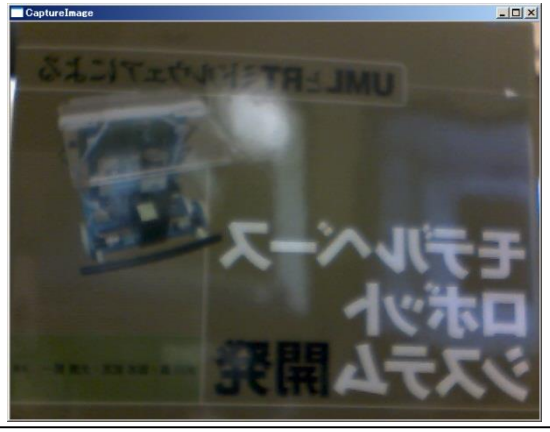


コンフィギュレーションの変更

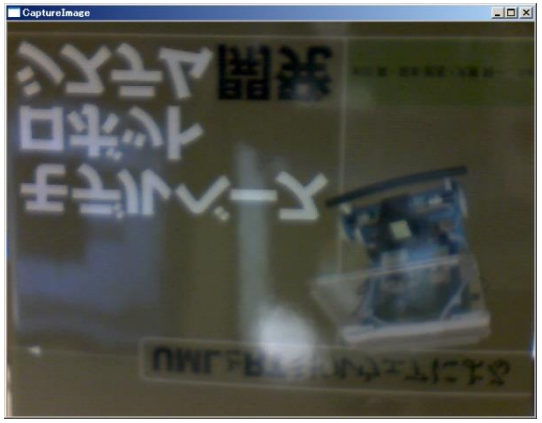


- ConfigurationViewの「編集」
- 表示されたダイアログ内で「flip_mode」の値を変更
- 「Apply」のチェックボックス

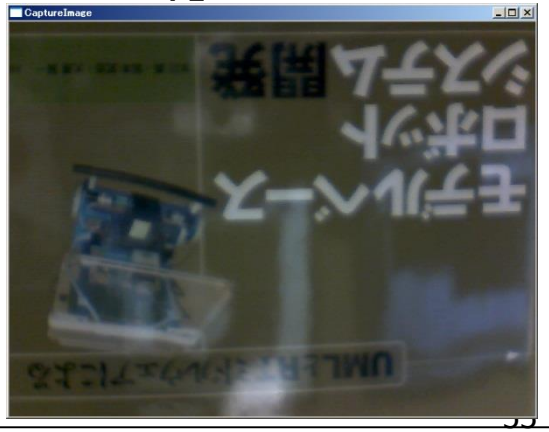
flip_mode=1



flip_mode=0

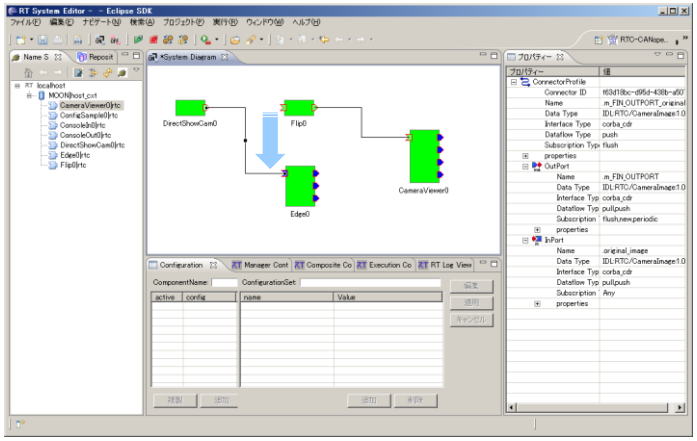


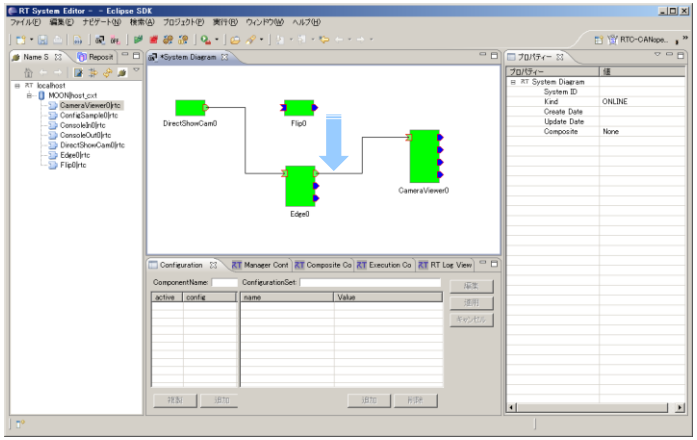
flip_mode=-1



システム構成の変更

- 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





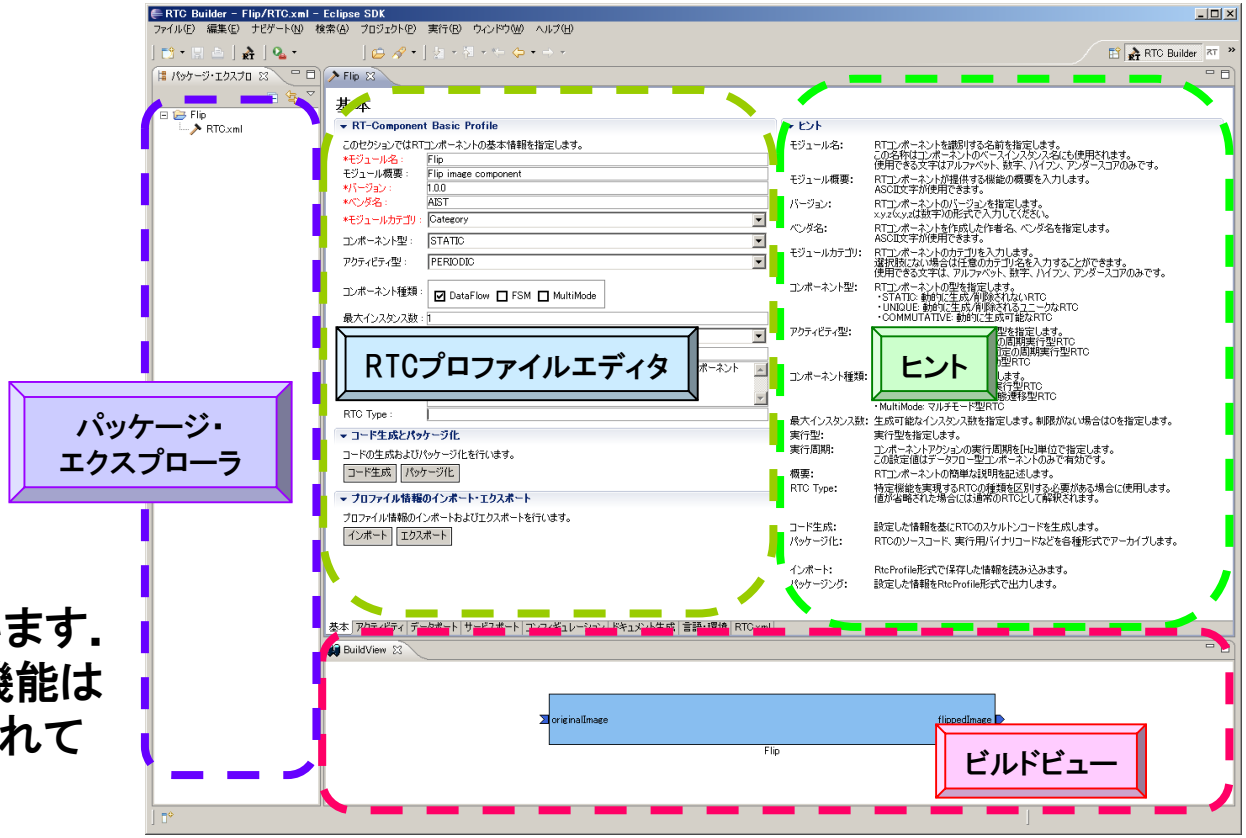
画像処理系のサンプルコンポーネント

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

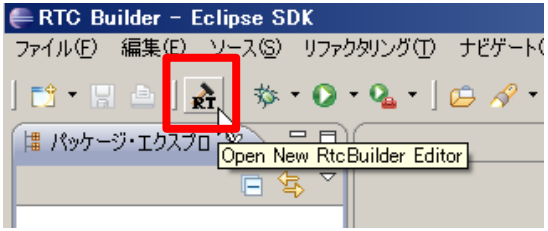
- RTCBuilderとは？
  - コンポーネントのプロファイル情報を入力し，ソースコード等の雛形を生成するツール
  - 開発言語用プラグインを追加することにより，各言語向けRTCの雛形を生成することが可能
    - C++
    - Java
    - Python



※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

プロジェクト作成/エディタ起動

①ツールバー内のアイコンをクリック

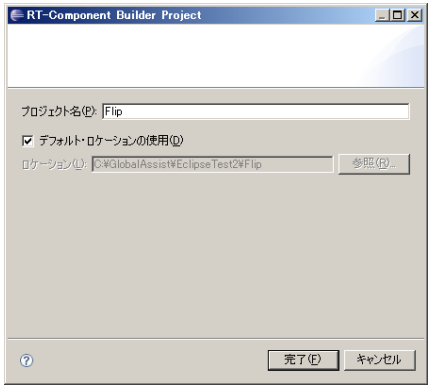


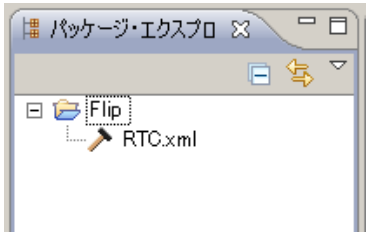
※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し，「次へ」
※メニューから「ファイル」-「Open New Builder Editor」を選択

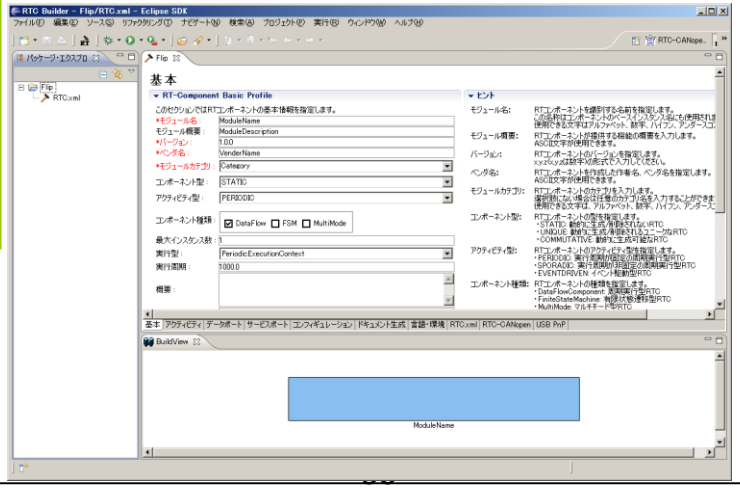
※任意の場所にプロジェクトを作成したい場合
②にて「デフォルト・ロケーションの使用」チェックボックスを外す
「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名：Flip

②「プロジェクト名」欄に入力し，「終了」

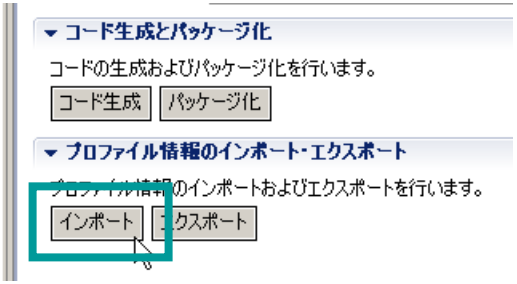




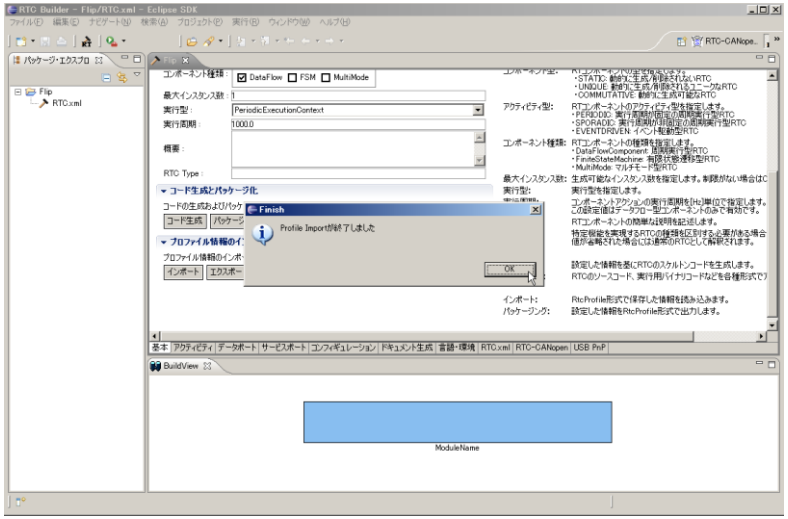


プロファイル インポート

①「基本」タブ下部の「インポート」ボタンをクリック



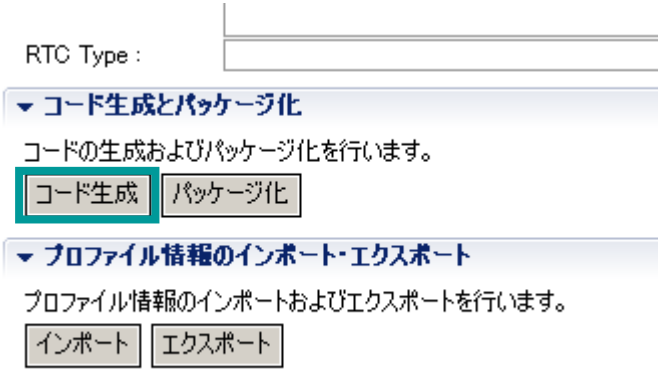
②【インポート】画面にて対象ファイルを選択

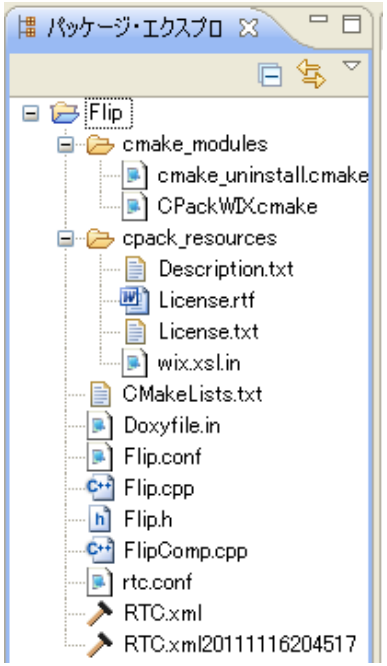


- 作成済みのRTコンポーネント情報を再利用
  - 「エクスポート」機能を利用して出力したファイルの読み込みが可能
  - コード生成時に作成されるRtcProfileの情報を読み込み可能
  - XML形式，YAML形式での入出力が可能

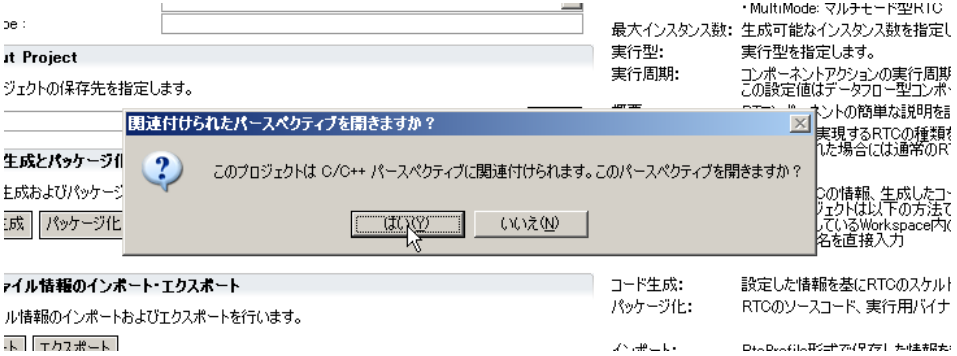
コード生成

- コード生成





- コード生成実行後，パースペクティブを自動切替



※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

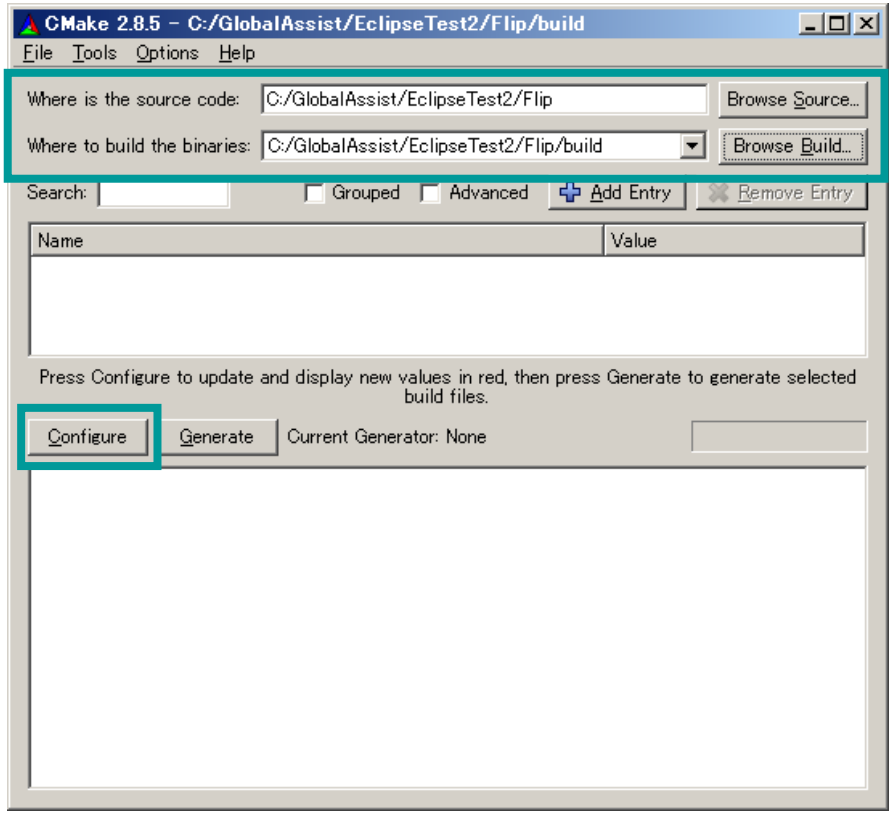
Java版RTC → JDT

（デフォルトインストール済み）

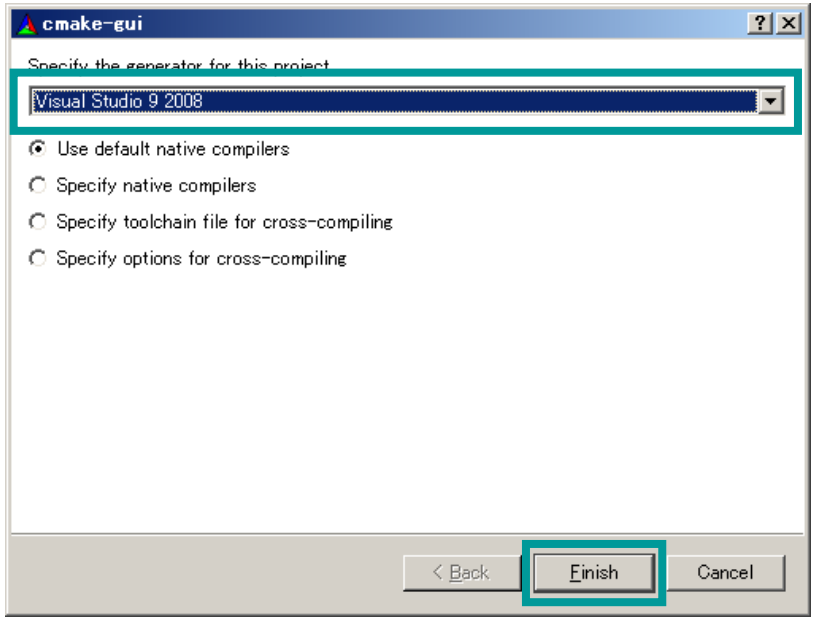
Python版 → PyDev

コンパイル(Windows,CMake利用)

①GUI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

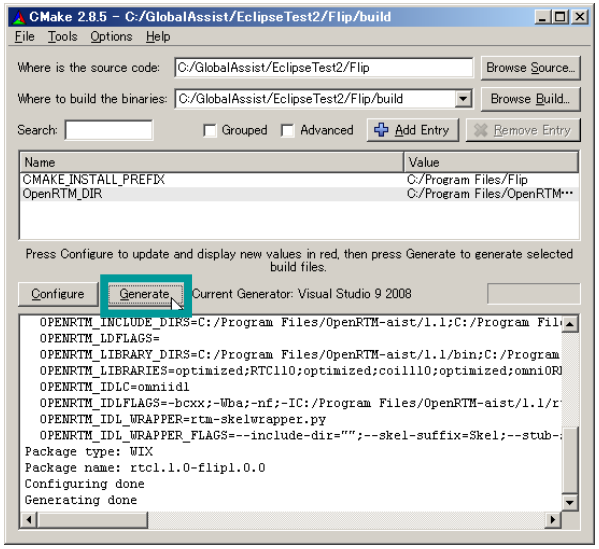


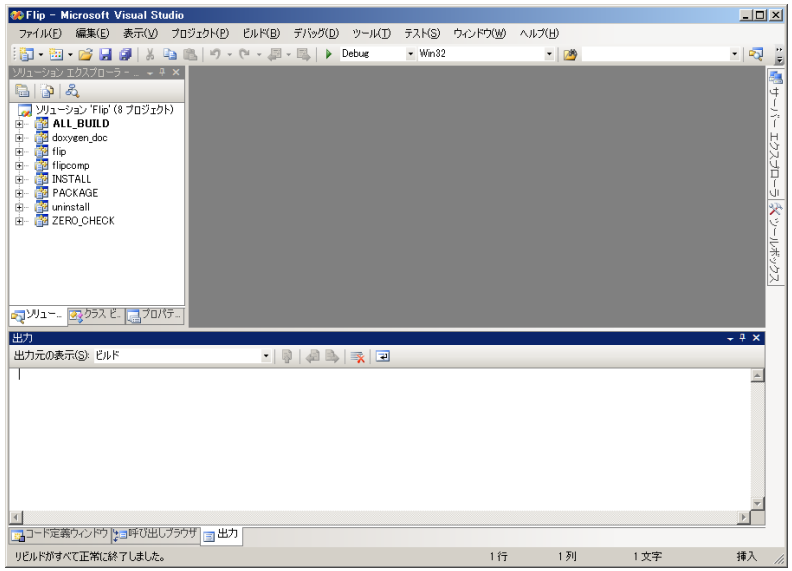
※binaryには，sourceとは別のディレクトリを指定する事を推奨

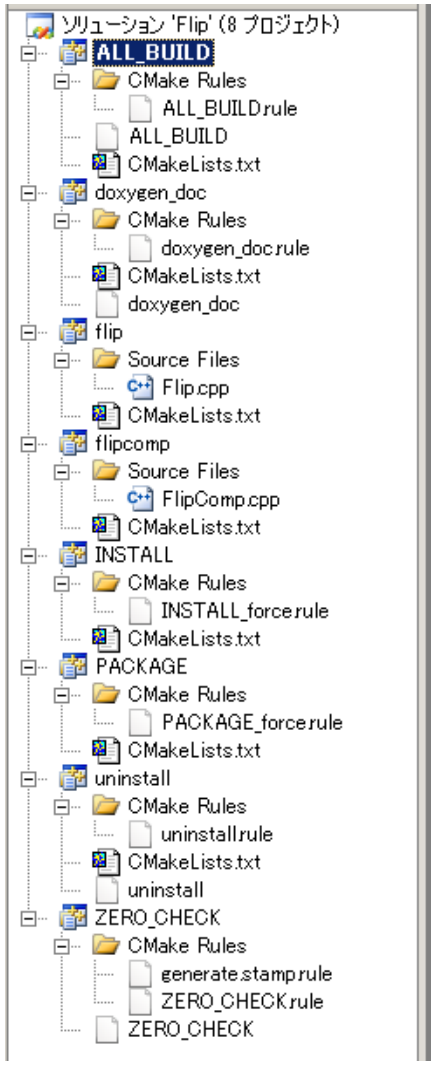
※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

コンパイル(Windows,CMake利用)

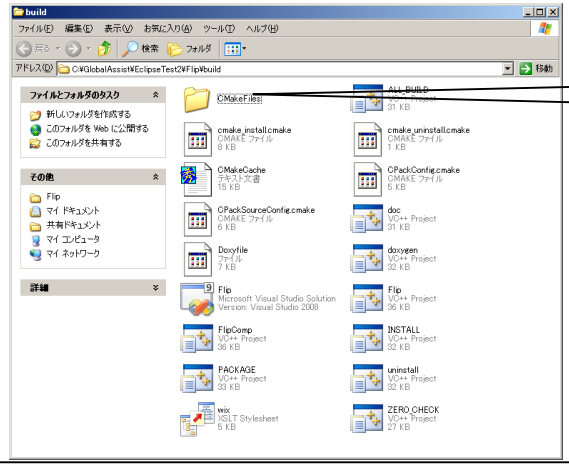
③正常終了後，「Generate」を実行







④binary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



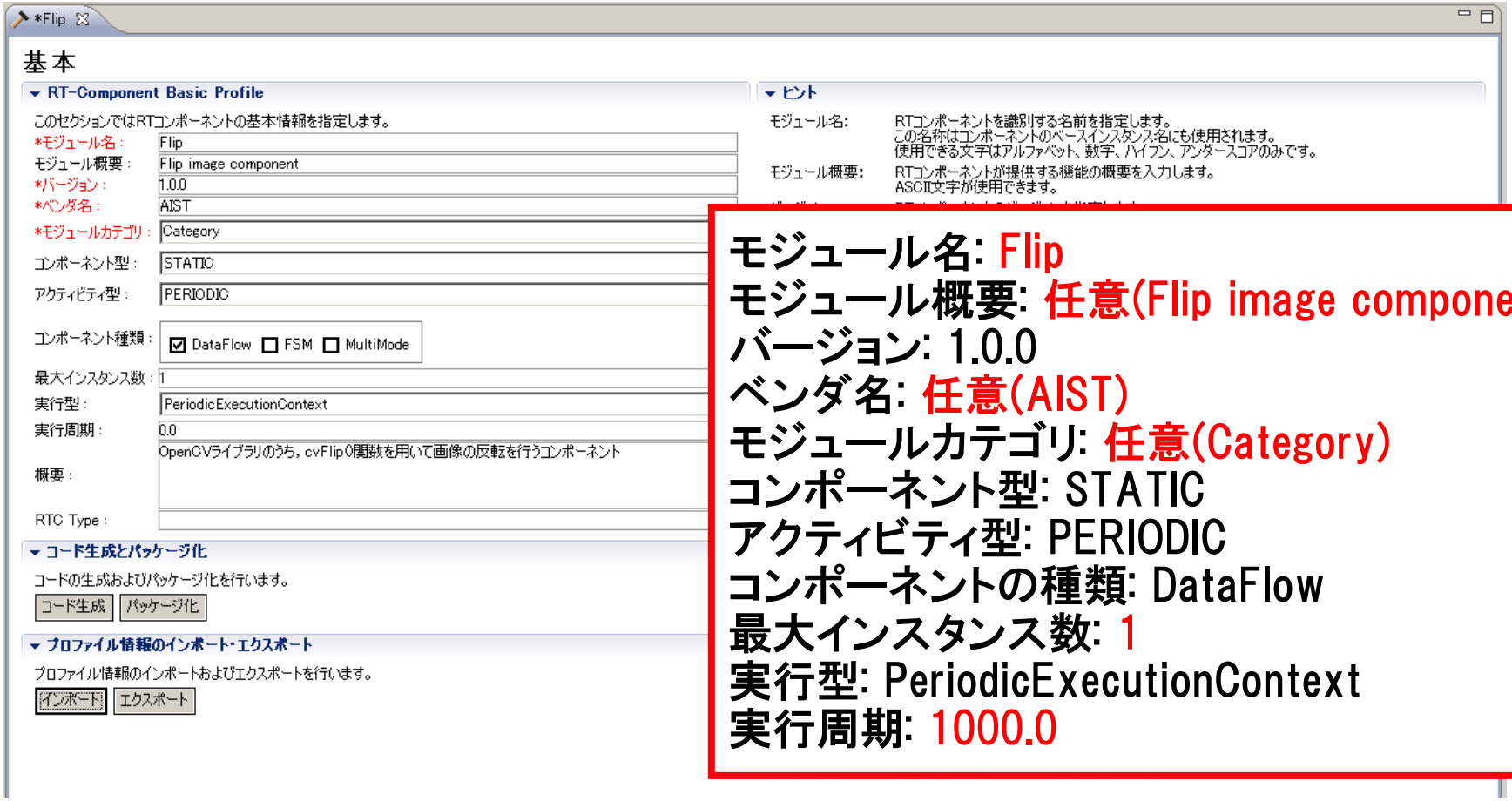
Flip
Microsoft Visual Studio Solution
Version: Visual Studio 2008

RTCプロファイルエディタ

画面要素名	説明
基本プロファイル	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・プロファイル	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・プロファイル	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・プロファイル	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

基本プロファイル

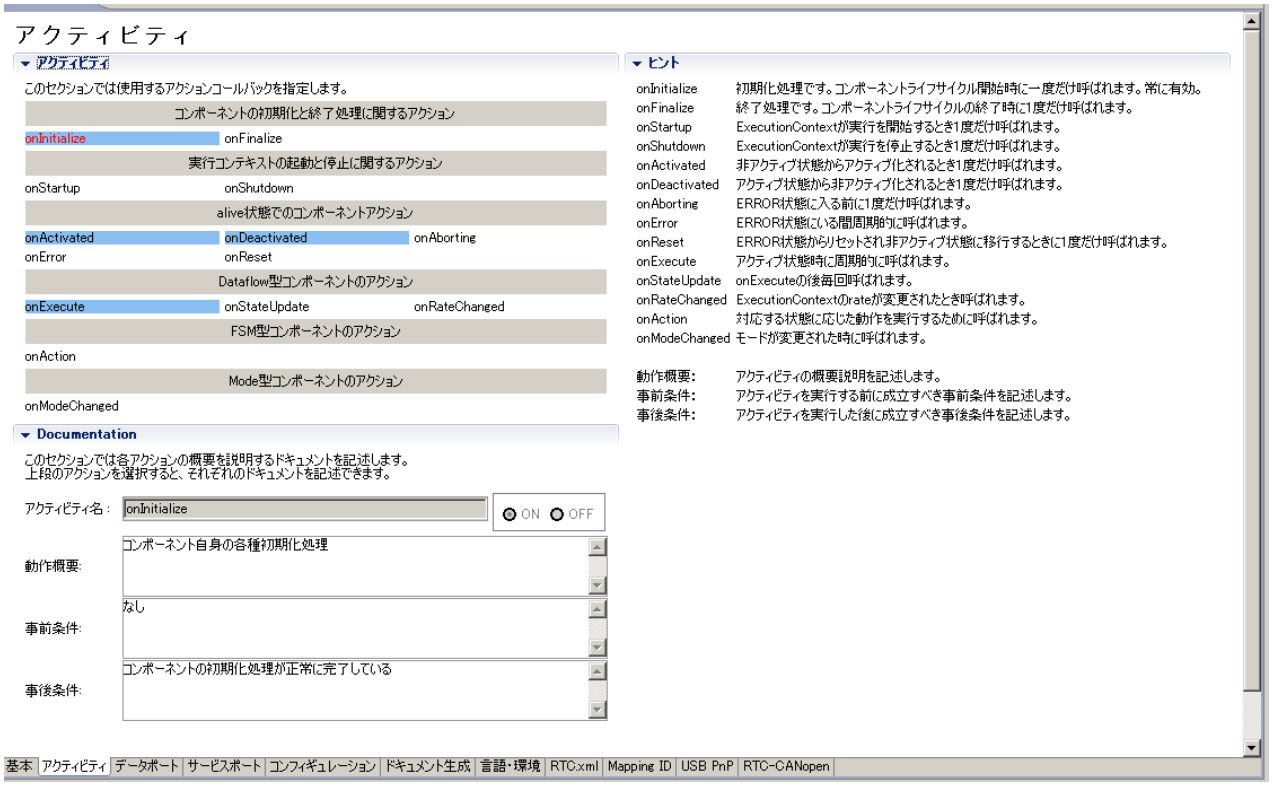
- RTコンポーネントの名称など，基本的な情報を設定



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

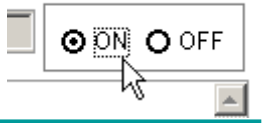
※エディタ内の項目名が赤字の要素は必須入力項目
※画面右側は各入力項目に関する説明

- 生成対象RTCで実装予定のアクティビティを設定



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

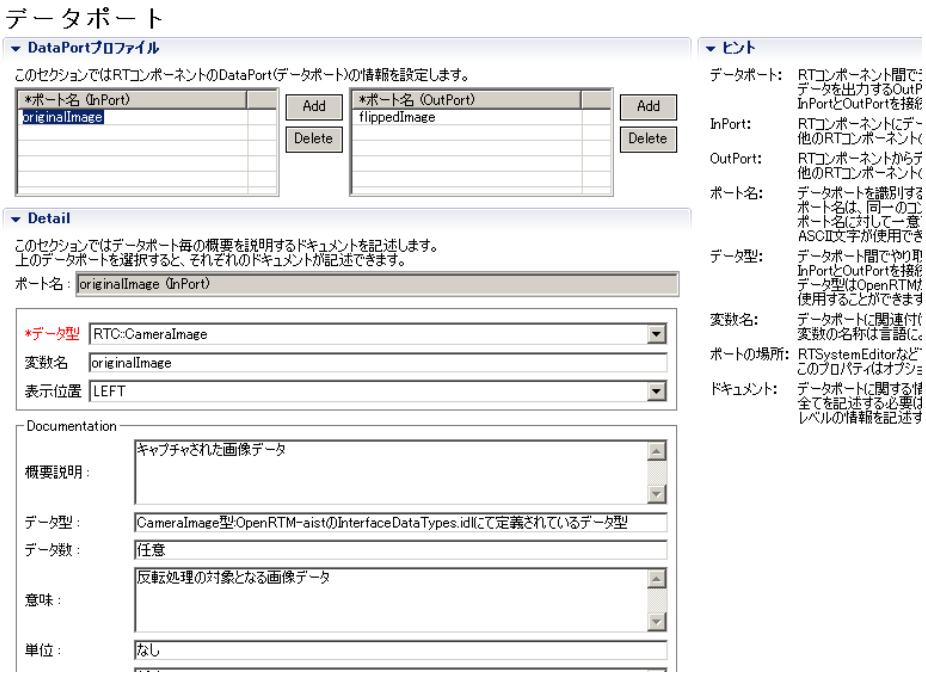


以下をチェック：
onActivated
onDeactivated
onExecute

※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

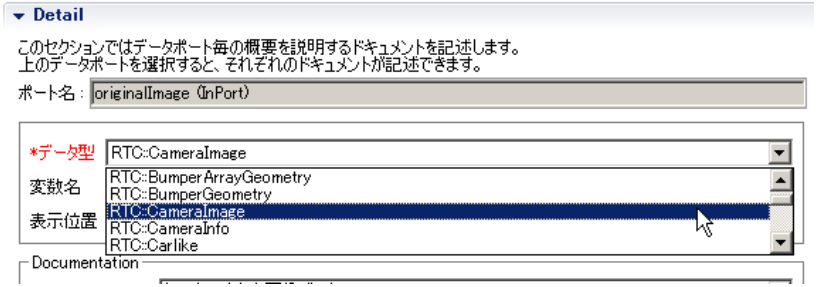
データポート・プロファイル

- 生成対象RTCに付加するDataPortの情報を設定



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

②設定する型情報を一覧から選択



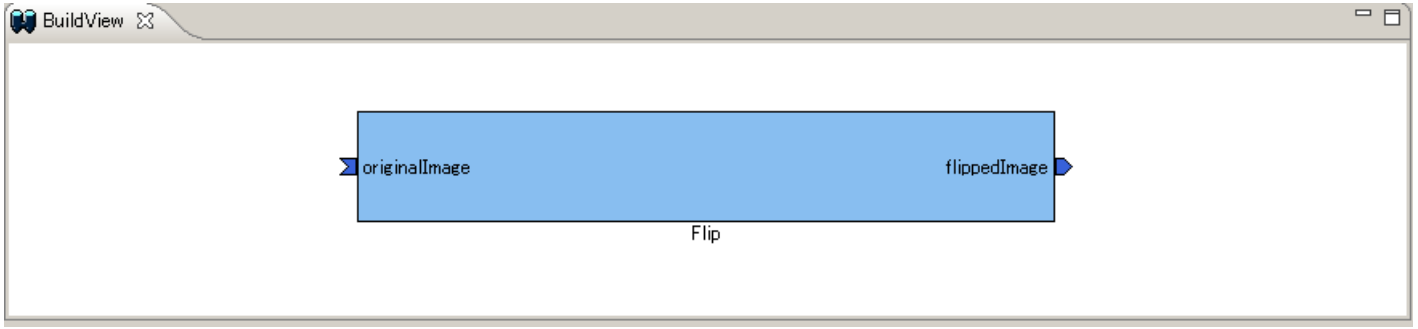
※データ型は，型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

※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

データポート・プロファイル

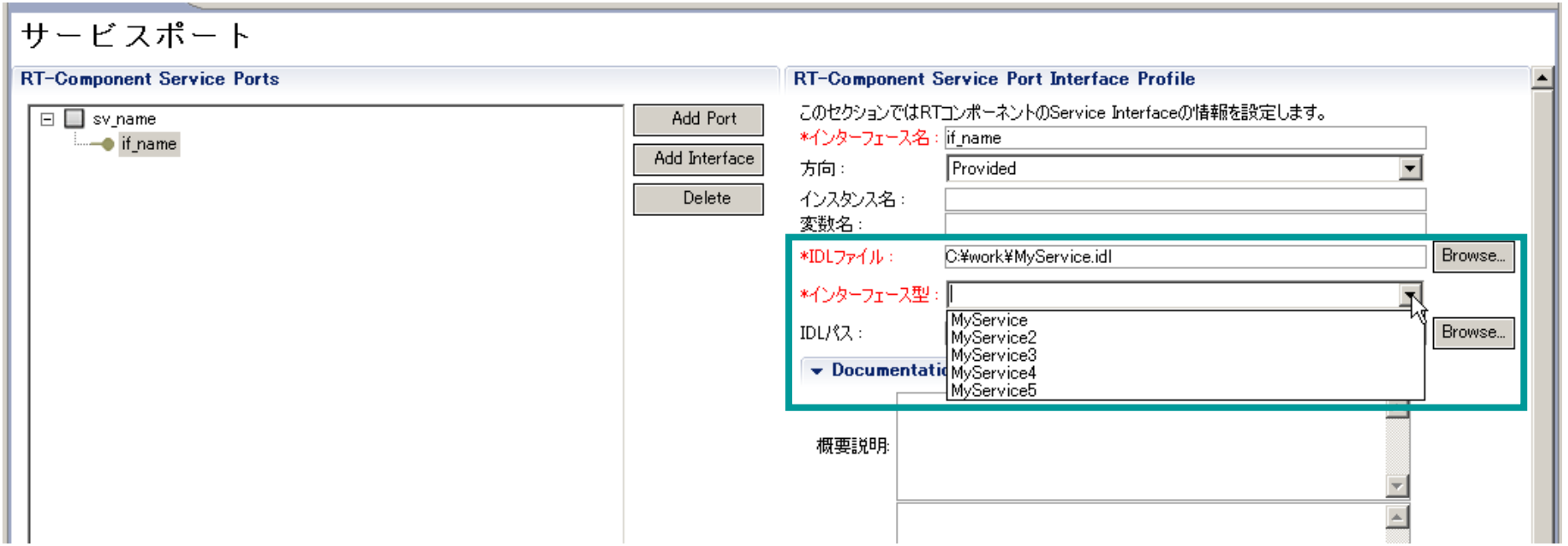
※Portの設定内容に応じて，下部のBuildViewの表示が変化



- 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

サービスポート・プロファイル

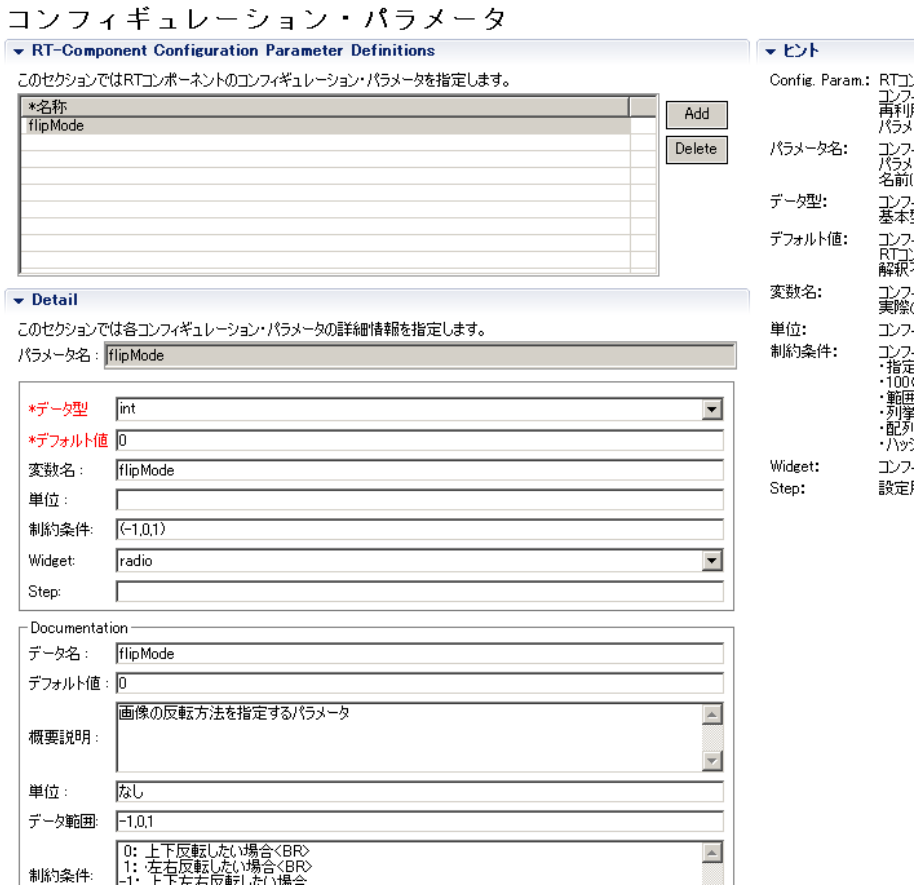
- 生成対象RTCに付加するServicePortの情報を設定



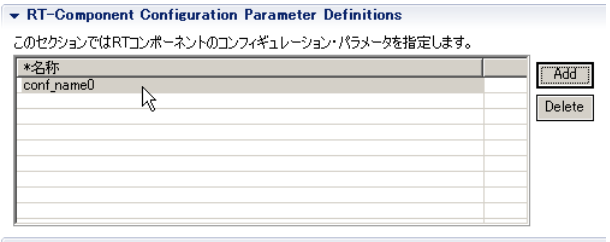
- サービスインターフェースの指定
  - IDLファイルを指定すると，定義されたインターフェース情報を表示

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



①「Add」ボタンをクリックし，追加後，直接入力で名称設定



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

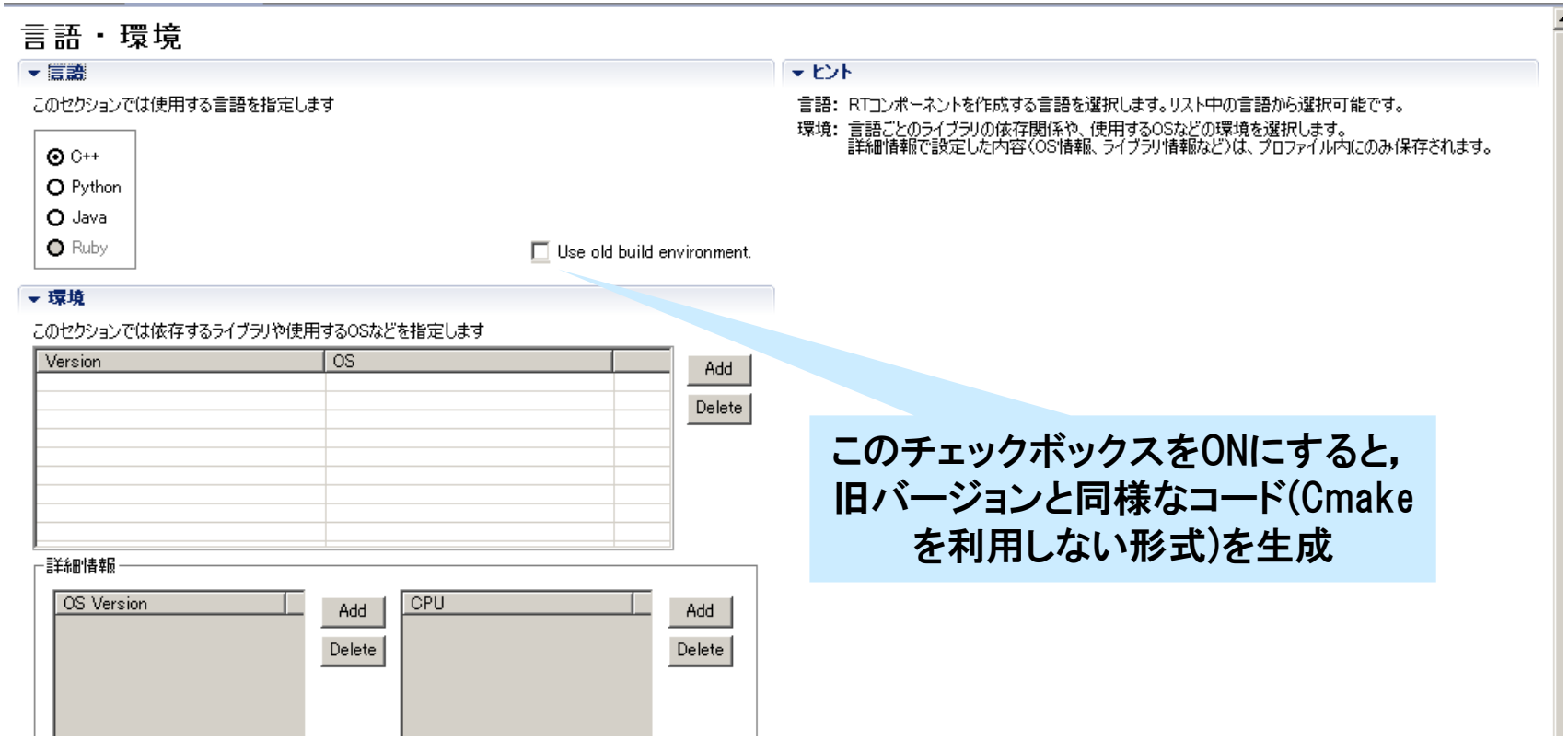
制約条件，Widgetの設定方法

- 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<=x<=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

言語・環境・プロファイル

- 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



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：Fip処理