

# RTミドルウェア講習会

---

日時:2012年7月19日(金) 13:00~17:00  
場所:名城大学 天白キャンパス 研究実験棟Ⅱ  
メカトロニクス工学科会議室



# OpenRT Platform



## ■ ロボット知能ソフトウェアプラットフォーム

- <http://www.openrtp.jp/wiki/>
- システム設計, シミュレーション, 動作生成, シナリオ生成などをサポート

## ■ OpenRT Platformツール群

- コンポーネント開発, システム開発における各開発フェーズの作業支援
- 開発プラットフォームにEclipseを採用

## ■ 構成

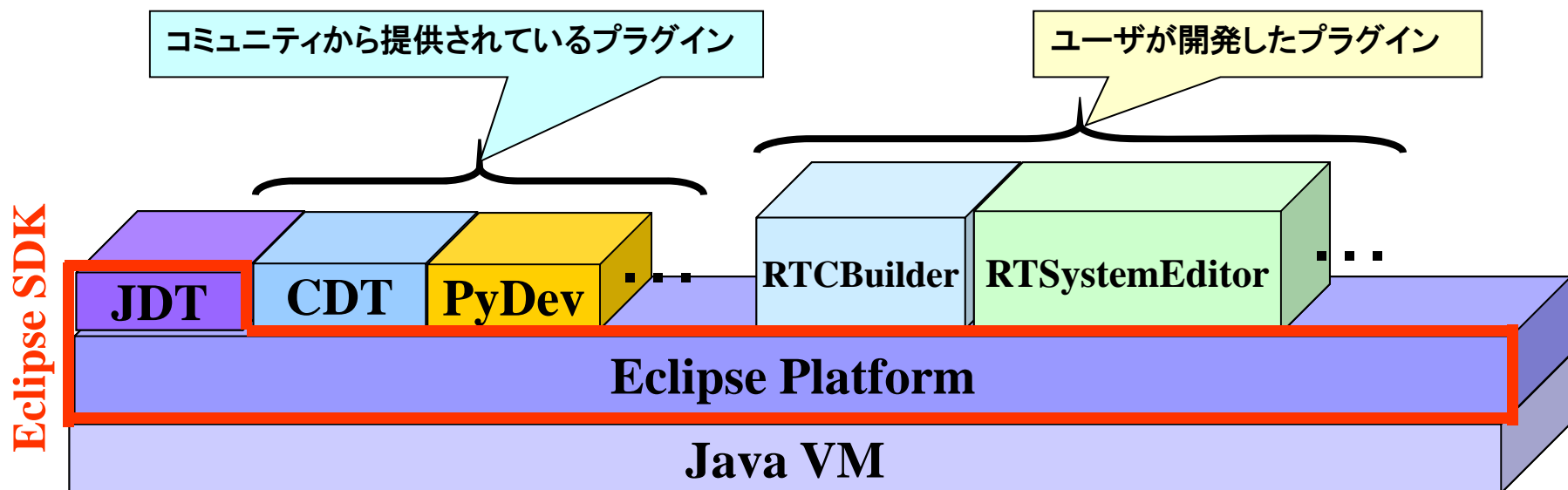
- RTCBिल्ダ
- RTCデバッガ
- RTシステムエディタ
- ロボット設計支援ツール
- シミュレータ
- 動作設計ツール
- シナリオ作成ツール  
など



# 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



# RTCBuilder, RTSystemEditorのインストール



## ■ ダウンロードし、解凍するだけ

※Javaの実行環境については、別途インストールが必要

The screenshot shows the OpenRTM-aist website. The main content area is titled "OpenRTM Eclipse tools 1.1.0-RC2". It includes a "Table of contents" section with links for "全部入りパッケージ", "バイナリ", "Eclipse/JDK/JRE等", and "過去のバージョン". Below this is a section for "全部入りパッケージ" which contains a table for "Eclipse-3.4.2 [Ganymede SR2]".

Eclipse-3.4.2 [Ganymede SR2]		
Eclipse3.4.2+RTSE+RTCB Windows用全部入り	<a href="#">eclipse342_rtmtools110-rc2_win32_ja.zip</a> MD5:2e6f9fa3e370b6e7ac1f9340d36c7abf	2011.07.22

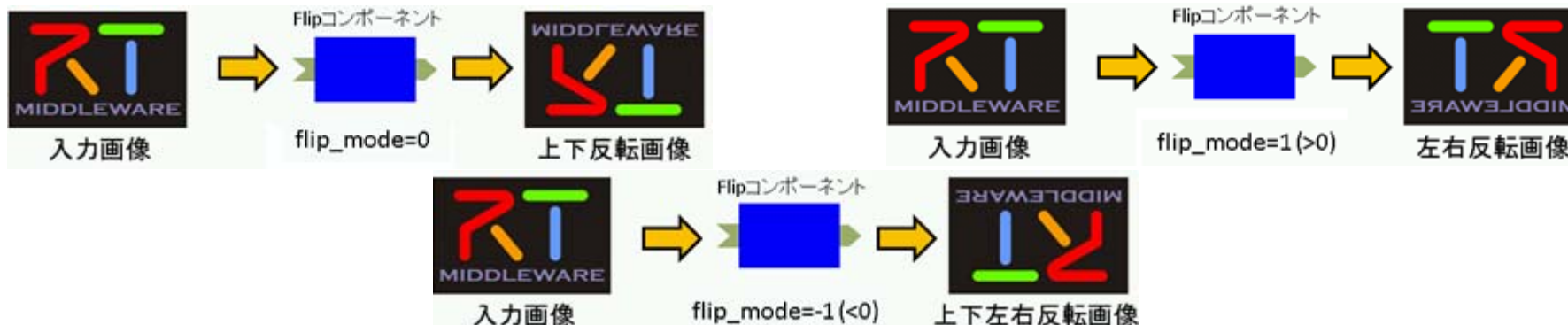
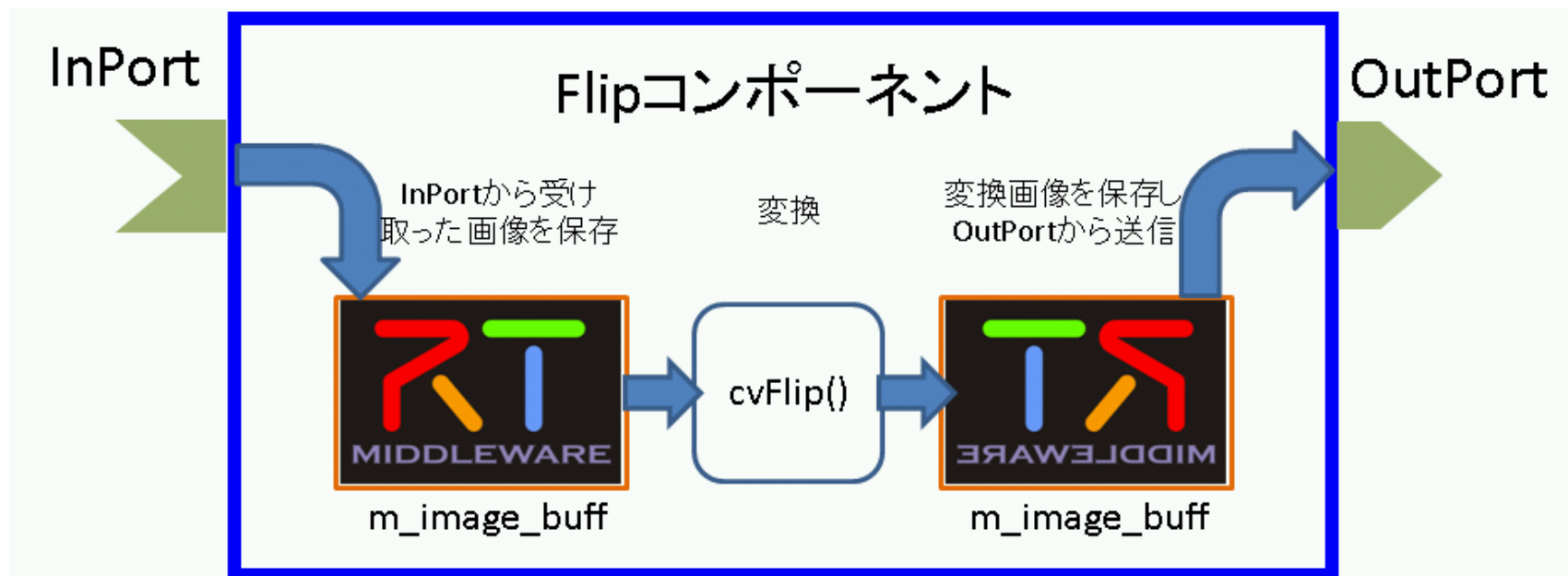
Below the table, there are instructions for Ubuntu users:

- Ubuntu8.04, Ubuntu9.10, Ubuntu10.04でLinux用Eclipse3.4.2が動作しない不具合が報告されています。
- Ubuntu8.04では、`apt-get install xulrunner-1.9`としてxulrunnerをアップデートしてください。
- Ubuntu9.10, Ubuntu10.04では、以下の方法を利用するか、Eclipse3.3もしくは3.5をご利用ください。

```
$ su
# vi /etc/apt/source.list
1行追加 - deb http://jp.archive.ubuntu.com/ubuntu/ jaunty main restricted
# apt-get update
# apt-get install xulrunner-1.9
# dpkg -l |grep xulrunner-1.9
ii xulrunner-1.9 1.9.0.8+nobinonly-0ubuntu2 XIII + XPCOM application runner
```

# 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



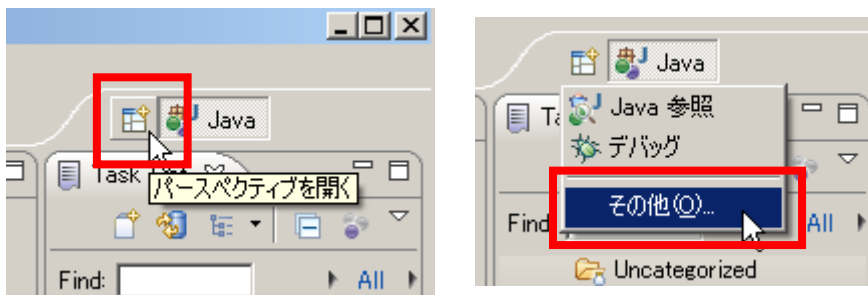
# システム構築支援ツール RTSystemEditorについて

---

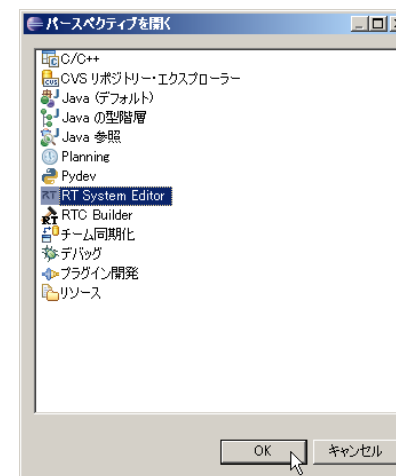


## ■ パースペクティブの切り替え

- 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

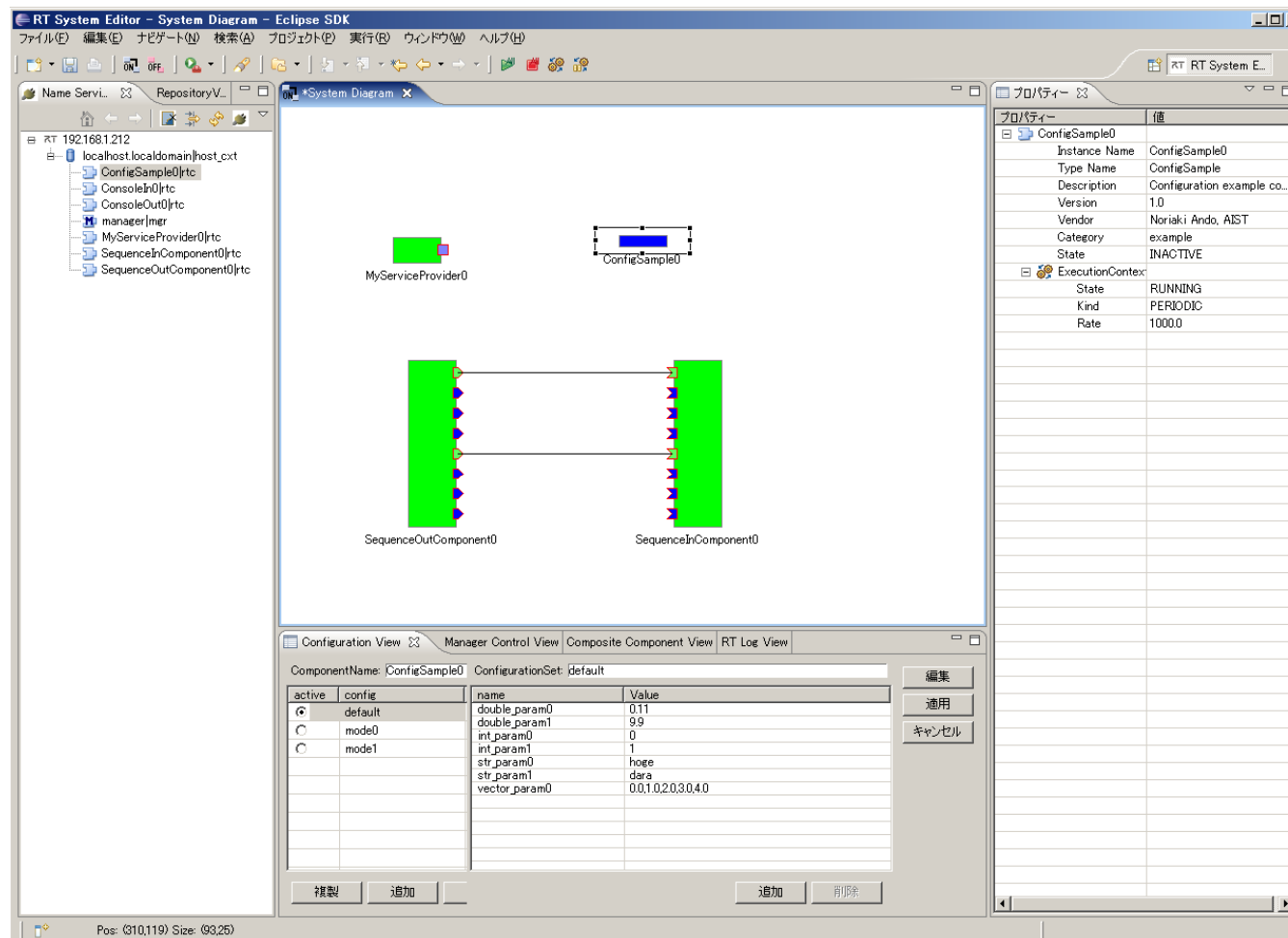


**※パースペクティブ**  
Eclipse上でツールの構成を管理する単位  
メニュー、ツールバー、エディタ、ビューなど  
使用目的に応じて組み合わせる  
独自の構成を登録することも可能

# 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





# 画面構成

The screenshot shows the RT System Editor interface. The main area displays a system diagram with components like MyServiceProvider0, ConfigSample0, and SequenceInComponent0. A pink box labeled 'システムエディタ' (System Editor) is overlaid on the diagram. To the left, a tree view shows the project structure, with a pink box labeled 'ネームサービスビュー' (Name Service View) pointing to it. To the right, a 'プロパティビュー' (Property View) shows details for ConfigSample0. At the bottom, a 'コンフィギュレーションビュー' (Configuration View) shows a table of configuration parameters.

ComponentName	ConfigSet	name	Value
ConfigSample0	default	double_param0	0.11
		double_param1	99
		int_param0	0
		int_param1	1
		str_param0	hoge

ネームサービスビュー

プロパティビュー

コンフィギュレーションビュー

This section shows several smaller screenshots of the RT System Editor. On the left, a 'マネージャビュー' (Manager View) shows module management options like 'Loadable Modules', 'Loaded Modules', and 'Active Components'. In the center, a '複合コンポーネントビュー' (Composite Component View) shows a table of components and their ports.

component	port
ConsoleIn0	ConsoleIn0.out
ConsoleOut0	ConsoleOut0.in

On the right, an '実行コンテキストビュー' (Execution Context View) shows execution context details for ConsoleOut0, including rate (1000.2) and state (PERIODIC RUNNING). Below it, a 'ログビュー' (Log View) shows a table of log messages.

component	time	level	component	logger	message
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!

マネージャビュー

複合コンポーネントビュー

実行コンテキストビュー

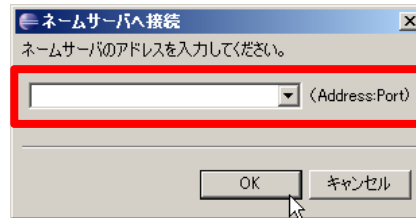
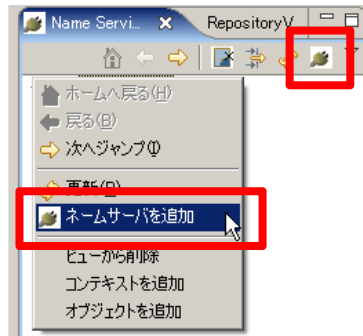
ログビュー

# カメラ関連コンポーネントの起動

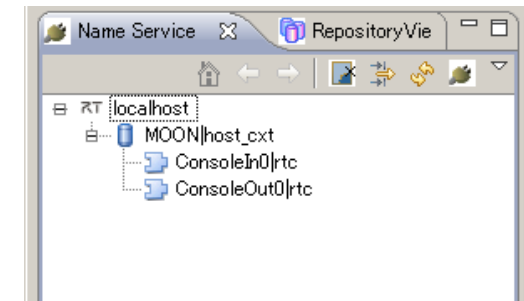
- Naming Serviceの起動
  - [スタート]メニューから  
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]
- CameraViewerCompの起動
  - [スタート]メニューから起動  
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [CameraViewerComp.exe]
- DirectShowCamCompの起動
  - [スタート]メニューから起動  
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]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

# RTシステム構築の基本操作

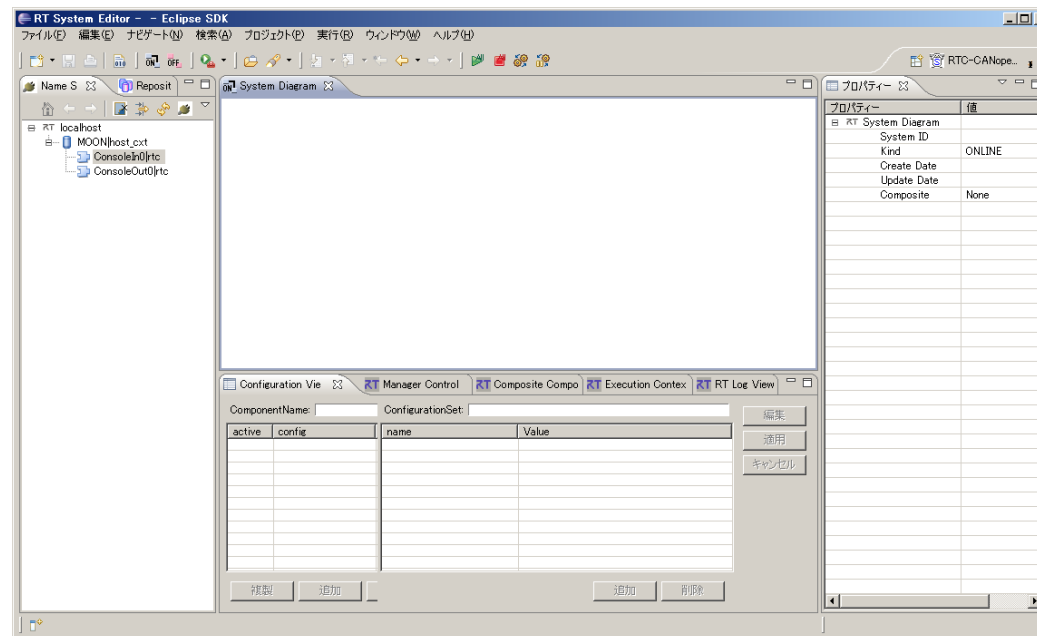
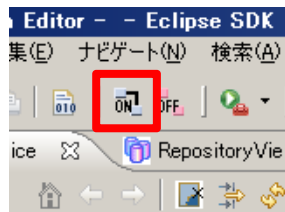
## ■ ネームサービスへ接続



※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

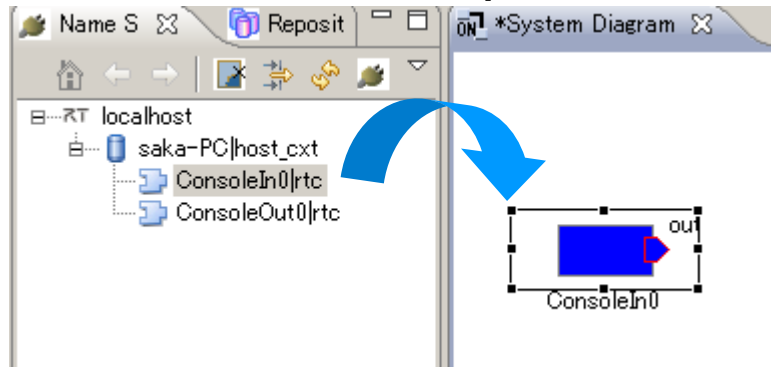


## ■ システムエディタの起動

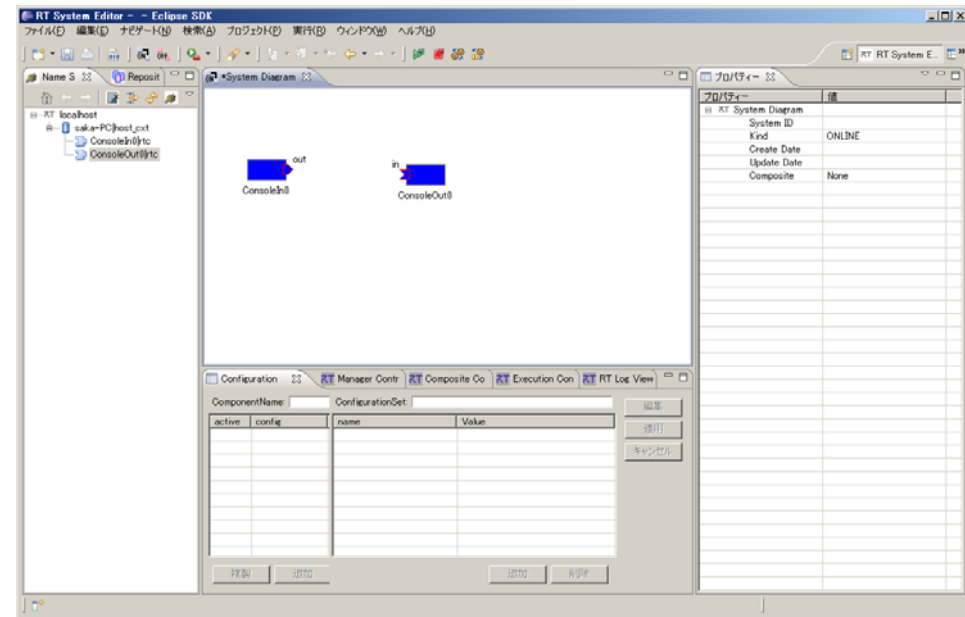


# RTシステム構築の基本操作

## RTコンポーネントの配置

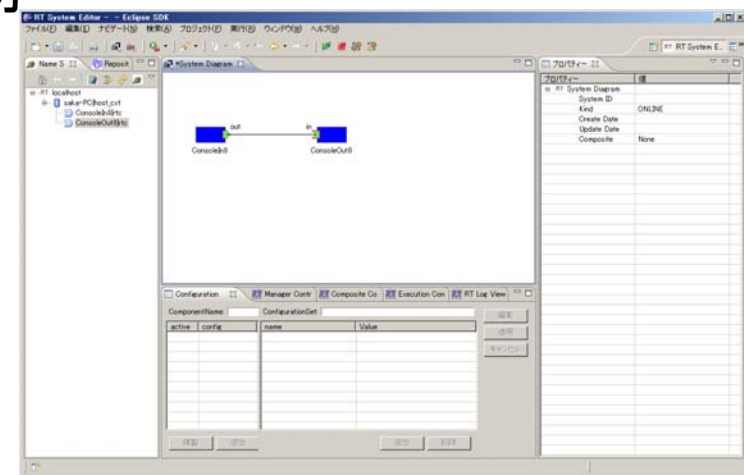
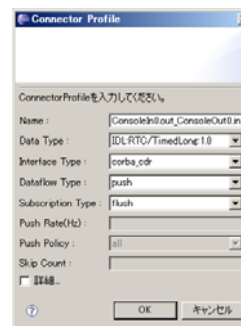
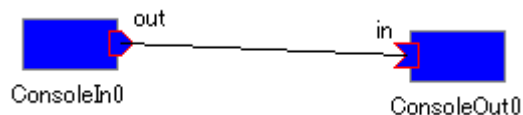


※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ



## ポートの接続

①接続元のポートから接続先の②接続プロファイルを入力ポートまでドラッグ



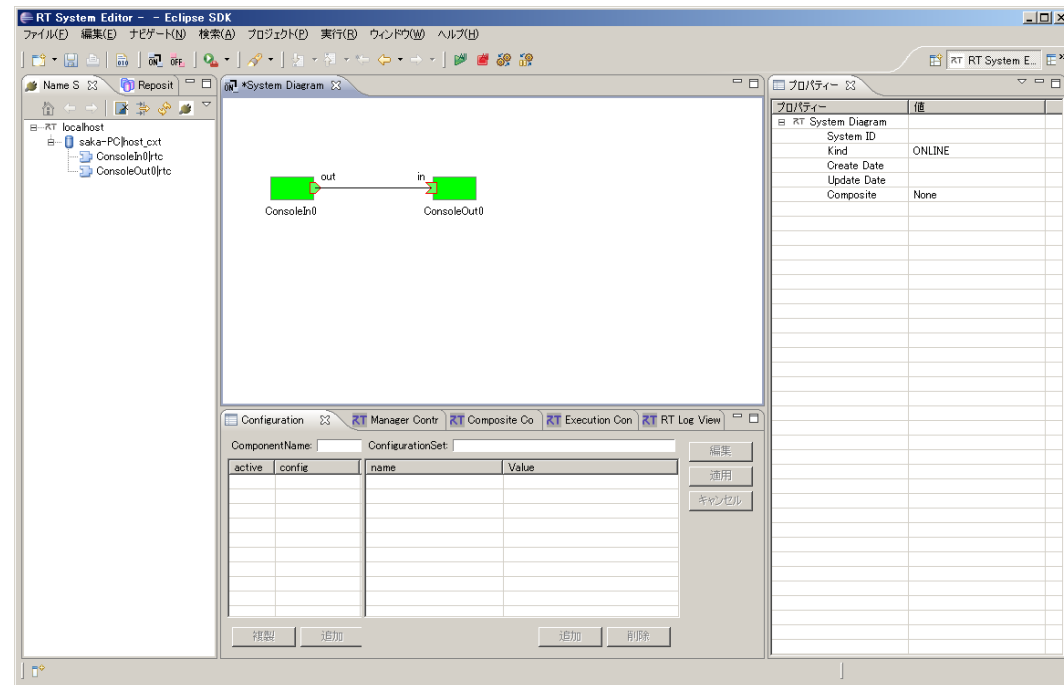
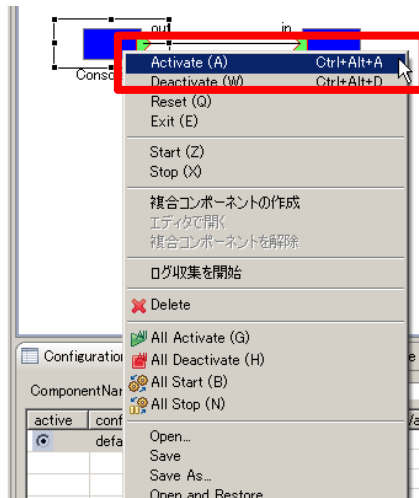
※ポートのプロパティが異なる場合など、接続不可能なポートの場合にはアイコンが変化



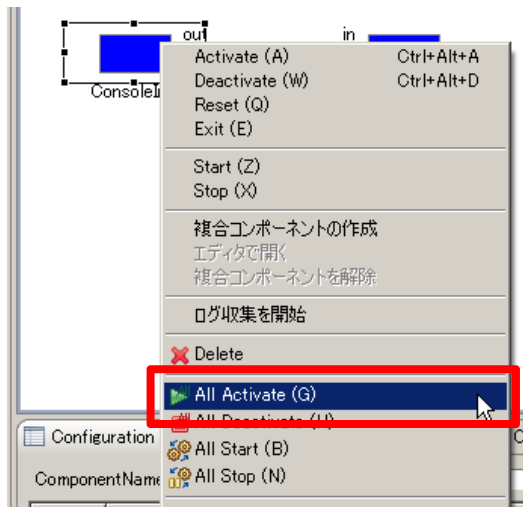
# RTシステム構築の基本操作

## ■ コンポーネントの起動

※各RTC単位で起動する場合



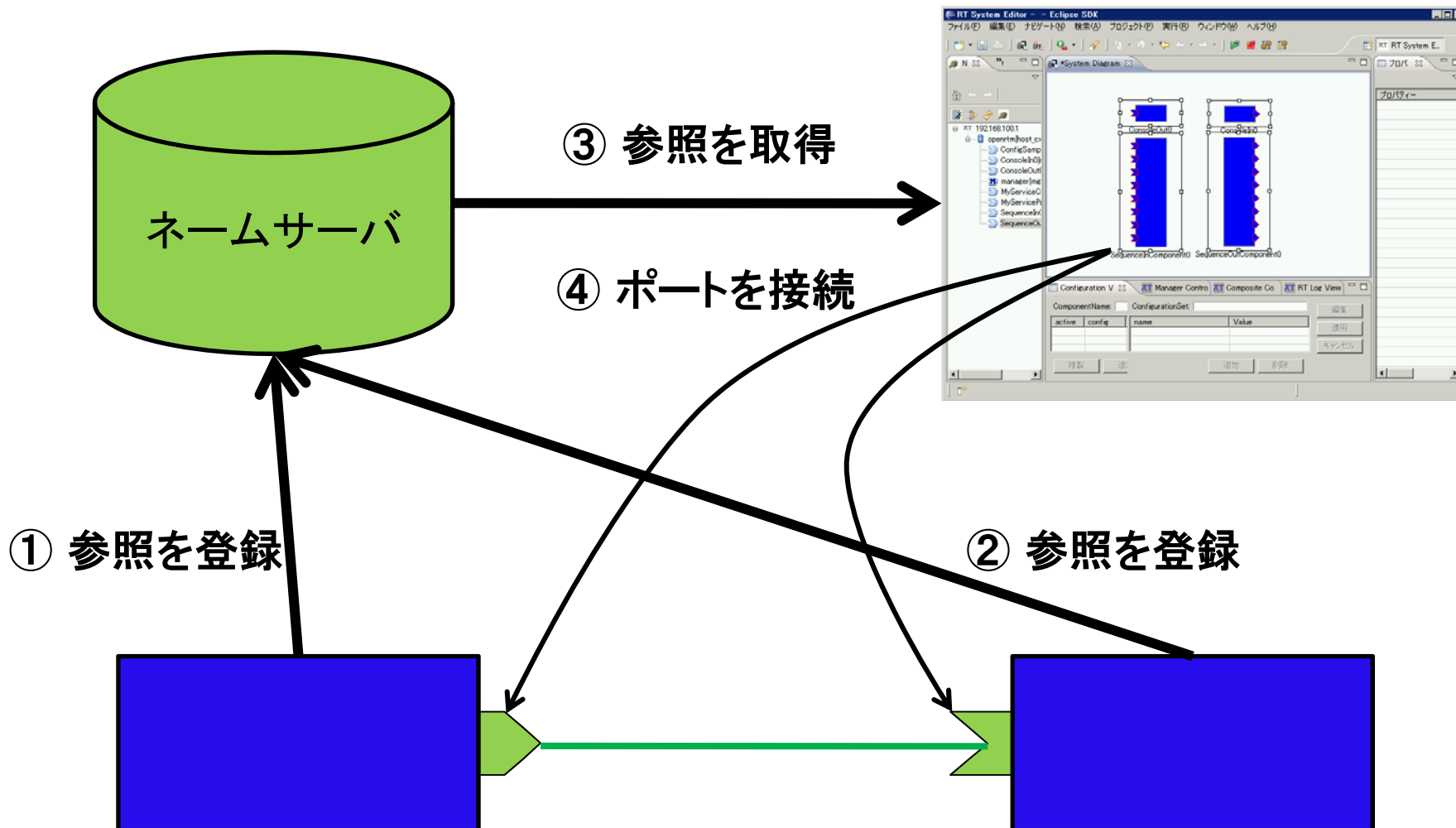
※全てのRTCを一括で起動する場合



※停止はDeactivateを実行

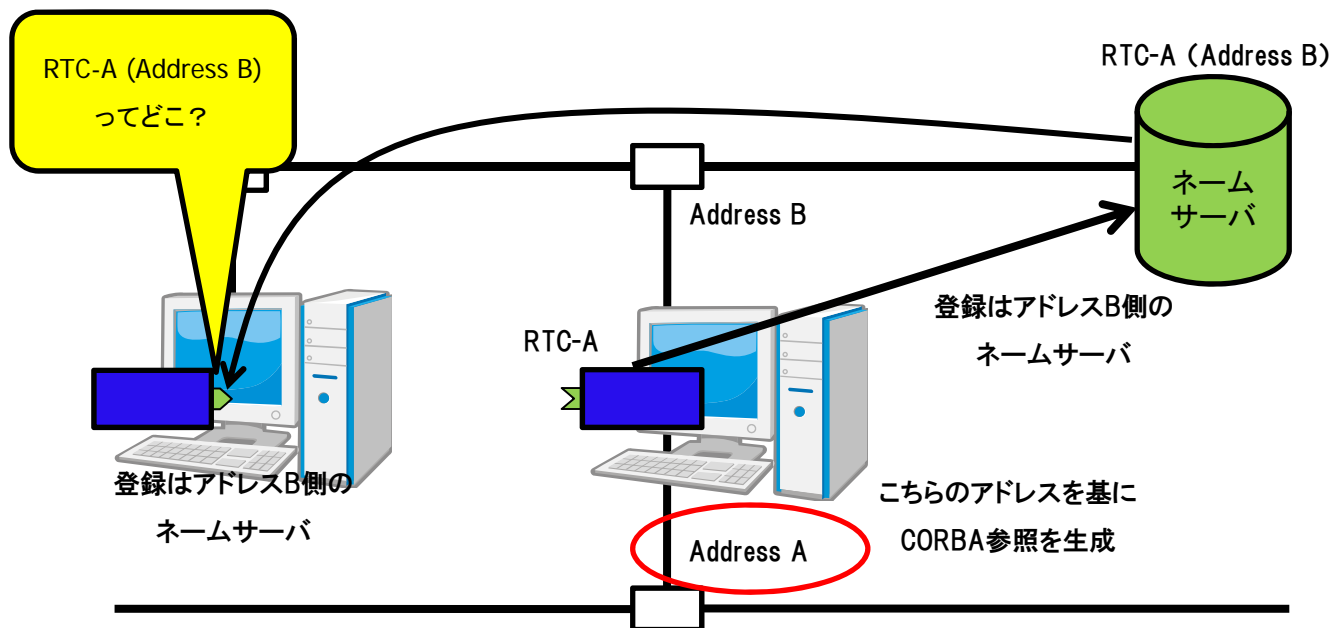
※RTC間の接続を切る場合には接続線をDelete  
もしくは、右クリックメニューから「Delete」を選択

# RTコンポーネントの動作シーケンス



# ネームサービスに接続できない場合

## ■ ネットワークインターフェースが2つある場合



## ■ RTC.confについて

- RTC起動時の登録先NamingServiceや、登録情報などについて記述

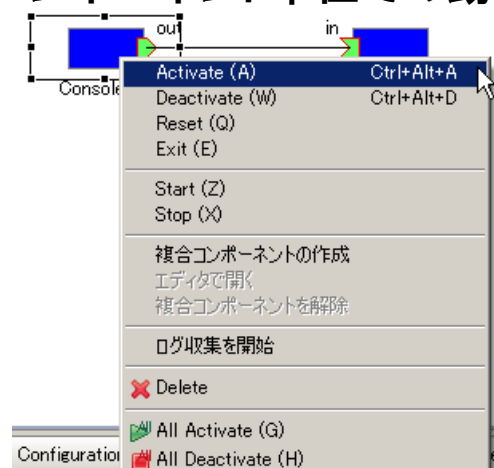
### ■ 記述例:

- **corba.nameservers**: localhost:9876
- **naming.formats**: SimpleComponent/%n.rtc
- **corba.endpoints**:192.168.0.12:

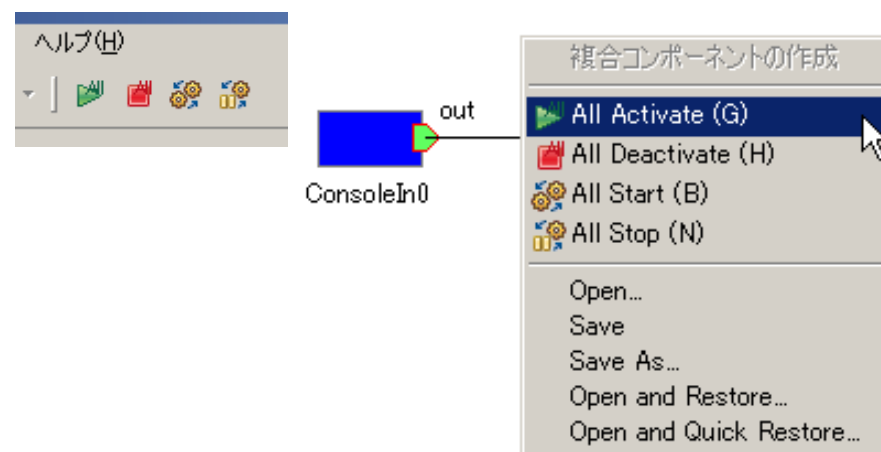
# RTコンポーネントの動作

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



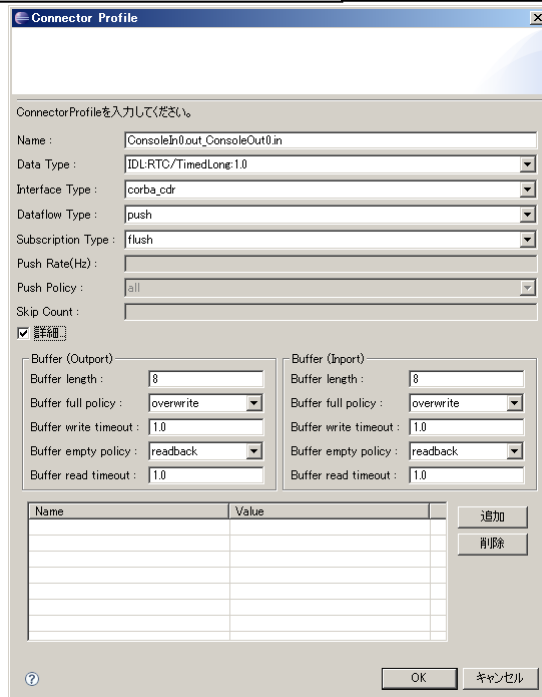
# 接続プロファイル(DataPort)について

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. <b>送信方法がPush</b> の場合有効. New, Periodic, Flushから選択
Push Rate	データ送信周期(単位:Hz). <b>SubscriptionTypeがPeriodic</b> の場合のみ有効
Push Policy	データ送信ポリシー. <b>SubscriptionTypeがNew, Periodic</b> 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. <b>Push PolicyがSkip</b> 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

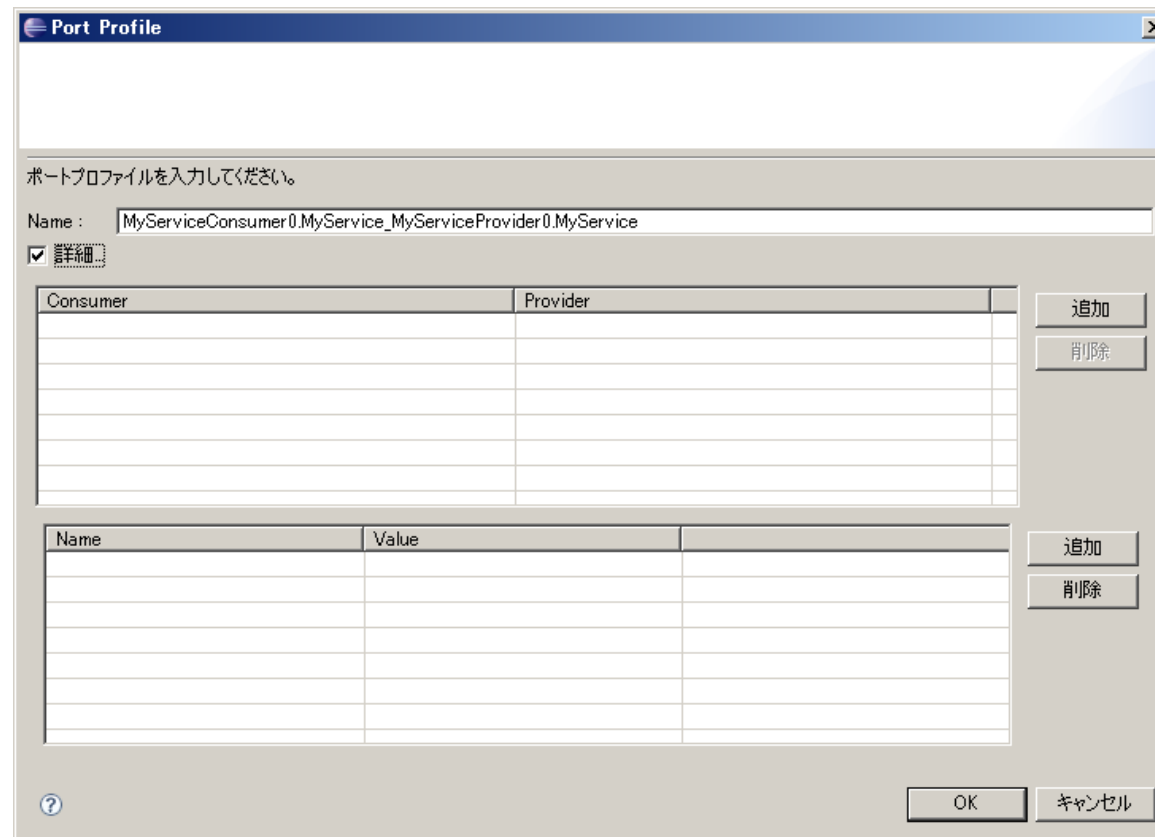
## ■ Buffer Policy

- 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

# 接続プロファイル(ServicePort)について

項目	設定内容
Name	接続の名称
インターフェース情報	接続するインターフェースを設定。 接続対象のServicePortに複数のServiceInterfaceが定義されていた場合、どのインターフェースを実際に接続するかを指定



Port Profile

ポートプロファイルを入力してください。

Name : MyServiceConsumer0.MyService\_MyServiceProvider0.MyService

詳細

Consumer	Provider

追加  
削除

Name	Value

追加  
削除

OK キャンセル

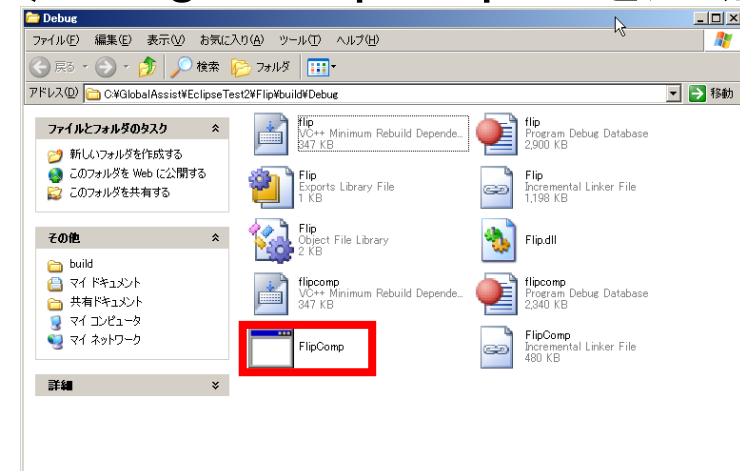
# 画像処理関連コンポーネントの起動

- 画像処理用コンポーネントの起動

- Flipコンポーネントの起動

先ほどコンパイルしたコンポーネントの起動

binaryにて指定したディレクトリ以下のSrc/Debug内のFlipComp.exeを起動



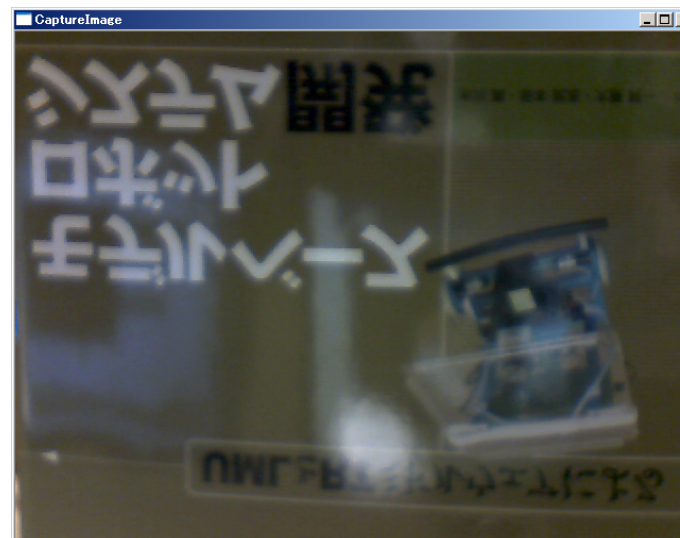
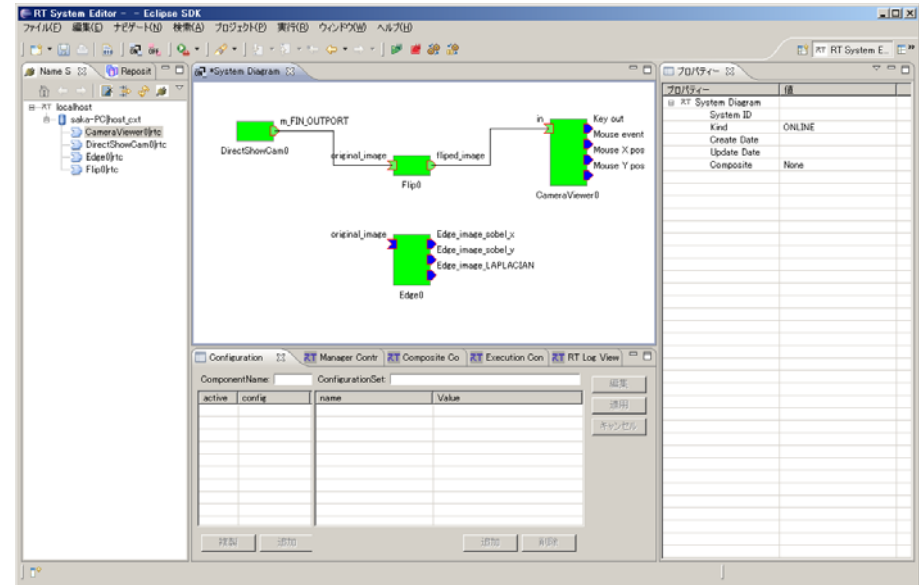
([プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [FlipComp.exe])

- [スタート]メニューから起動

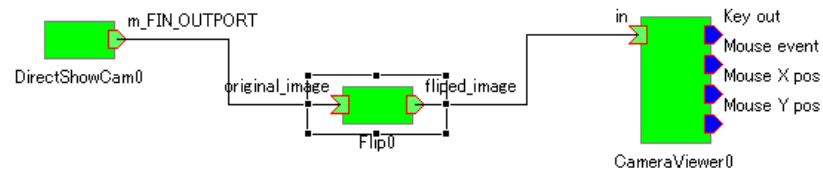
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]  
→[opencv-rtcs]→ [EdgeComp.exe]

# システムの構成

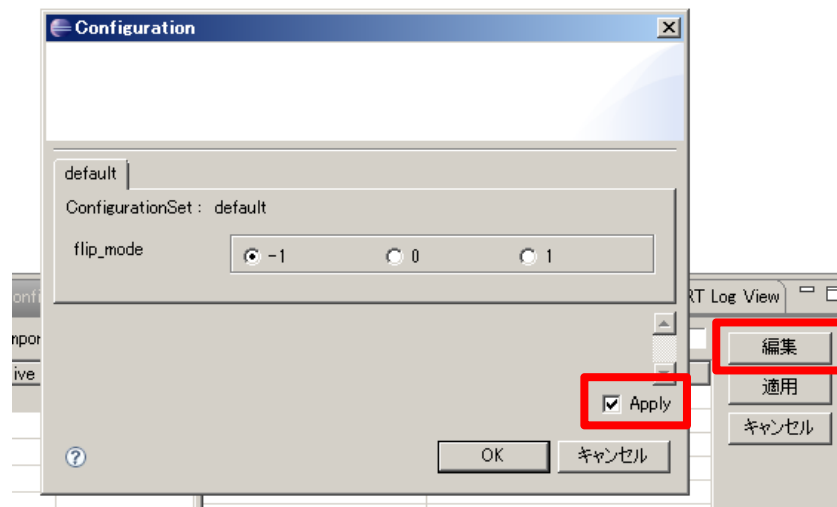
- Flip側との接続
  - DirectShowCam → Flip  
→ CameraViewerと接続  
(接続プロファイルはデフォルト設定)
  - AllActivateを実行



# コンフィギュレーションの変更



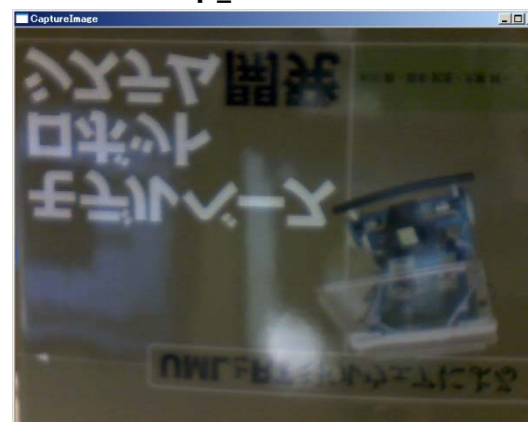
- ConfigurationViewの「編集」
- 表示されたダイアログ内で「flip\_mode」の値を変更
- 「Apply」のチェックボックス



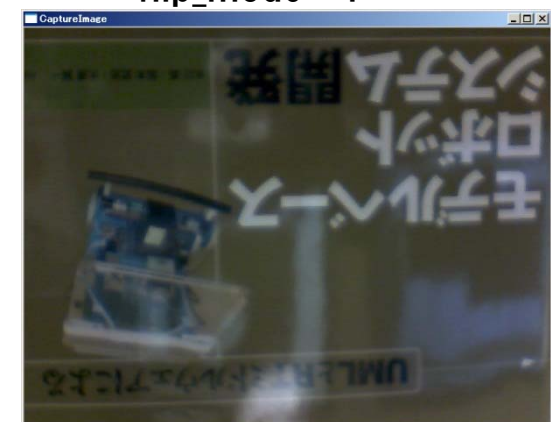
flip\_mode=1



flip\_mode=0

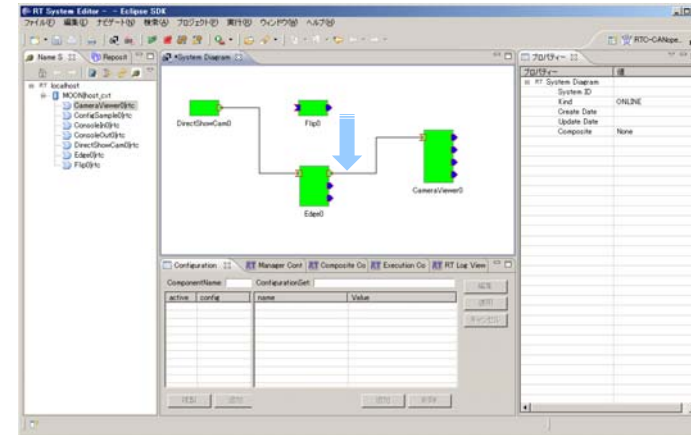
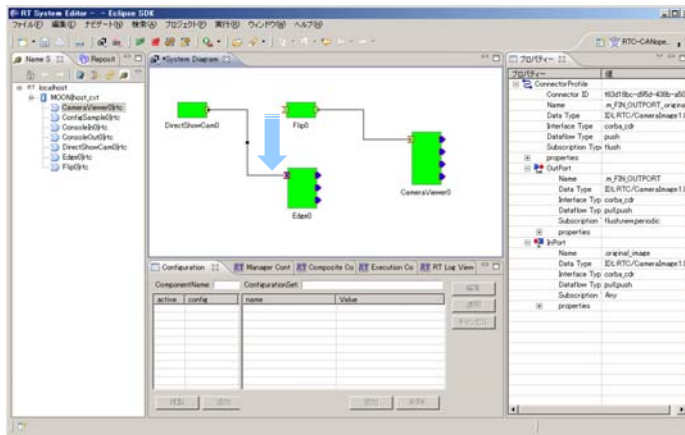


flip\_mode=-1



# 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



# コンポーネント開発ツール RTCBuilderについて

---





# RTCBuilder概要

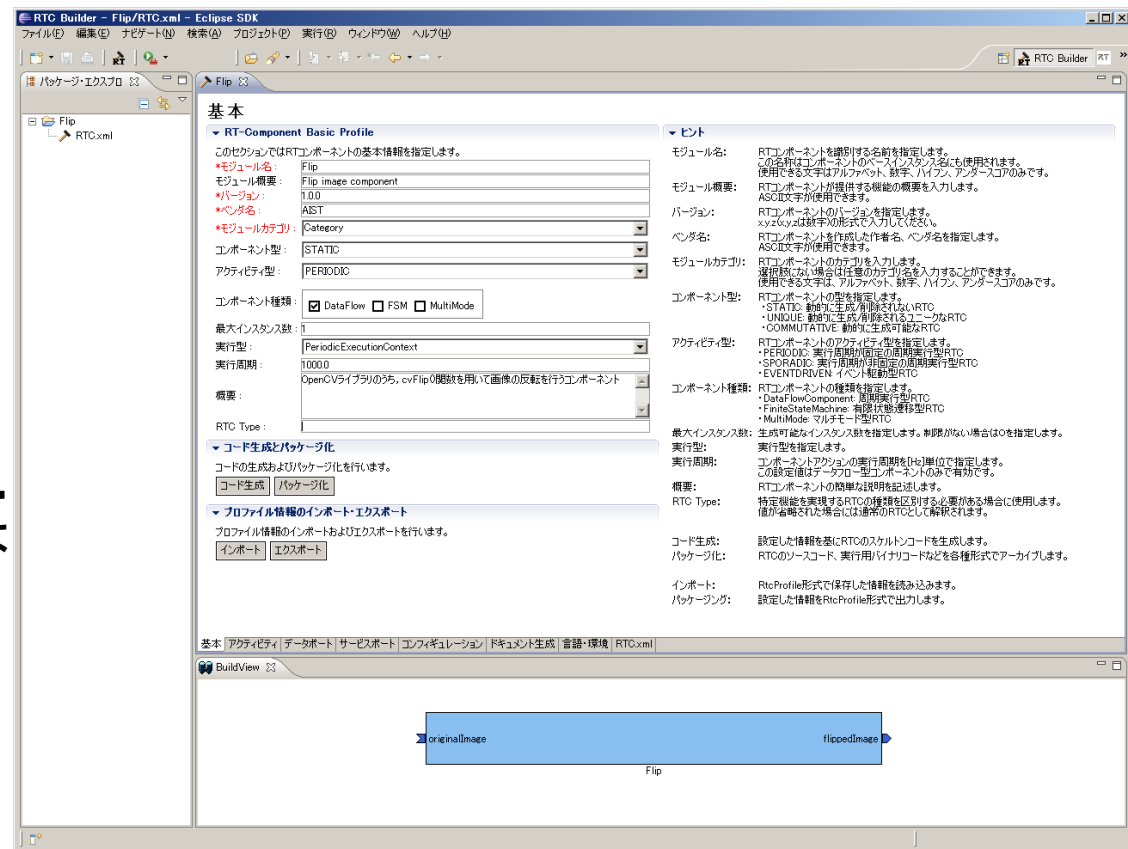


## ■ RTCBuilderとは？

- 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



# 画面構成

The screenshot shows the RTC Builder application window. On the left, the Package Explorer shows a project named 'Flip' with an 'RTC.xml' file. The main area is divided into several sections:

- 基本 (Basic):** Contains the 'RT-Component Basic Profile' section with fields for Module Name (Flip), Summary (Flip image component), Version (1.0.0), Vendor (AIST), Category (Category), Component Type (STATIC), and Activity Type (PERIODIC). It also has checkboxes for DataFlow, FSM, and MultiMode, and a field for Maximum Instance Count (1).
- ヒント (Hints):** A detailed section on the right providing instructions for each field in the Basic profile.
- コード生成とパッケージ化 (Code Generation and Packaging):** Includes buttons for 'コード生成' (Code Generation) and 'パッケージ化' (Packaging).
- プロファイル情報のインポート・エクスポート (Import/Export Profile Information):** Includes buttons for 'インポート' (Import) and 'エクスポート' (Export).

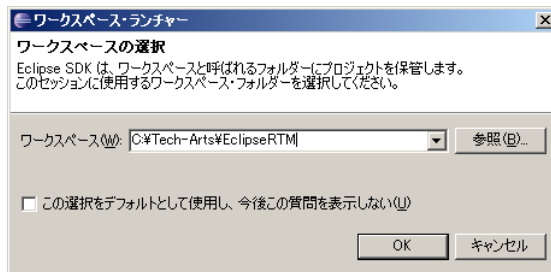
Callouts in the image:

- A purple box labeled 'パッケージ・エクスプローラ' (Package Explorer) points to the left sidebar.
- A blue box labeled 'RTCプロファイルエディタ' (RTC Profile Editor) points to the '基本' section.
- A green box labeled 'ヒント' (Hints) points to the 'ヒント' section.
- A pink box labeled 'ビルドビュー' (Build View) points to the bottom section showing a diagram with 'originalImage' and 'flippedImage' nodes connected by an arrow, with the label 'Flip' below.

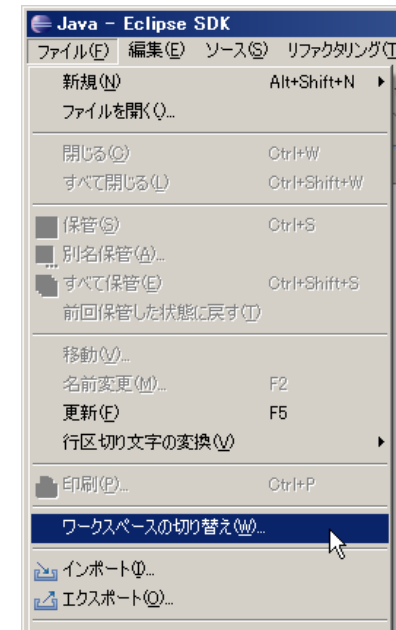
# ツールの起動

- Windowsの場合
  - Eclipse.exeをダブルクリック
- Unix系の場合
  - ターミナルを利用してコマンドラインから起動
    - Ex) \$ /usr/local/Eclipse/eclipse

## ■ ワークスペースの選択(初回起動時)



## ■ ワークスペースの切替(通常時)



### ※ワークスペース

Eclipseで開発を行う際の作業領域

Eclipse上でプロジェクトやファイルを作成するとワークスペースとして指定したディレクトリ以下に実際のディレクトリ、ファイルを作成する

- 初期画面のクローズ
  - 初回起動時のみ

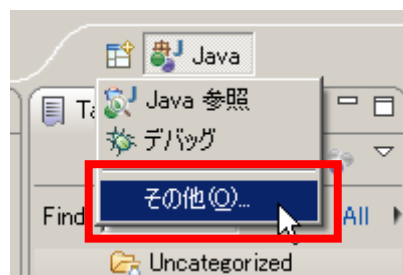
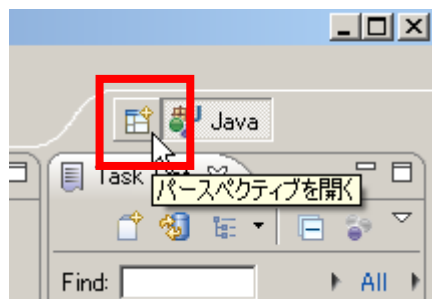


## ※パースペクティブ

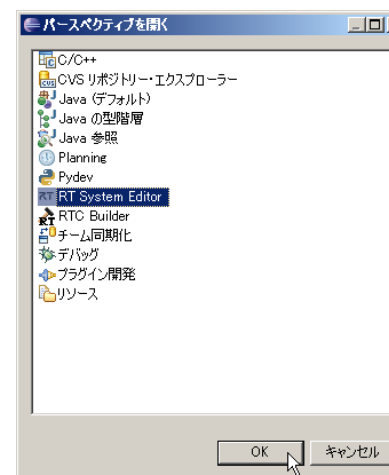
Eclipse上でツールの構成を管理する単位メニュー、ツールバー、エディタ、ビューなど使用目的に応じて組み合わせる独自の構成を登録することも可能

## ■ 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

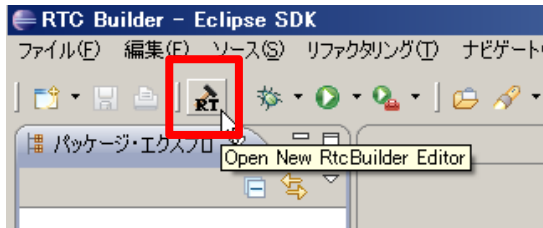


②一覧画面から対象ツールを選択



# 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

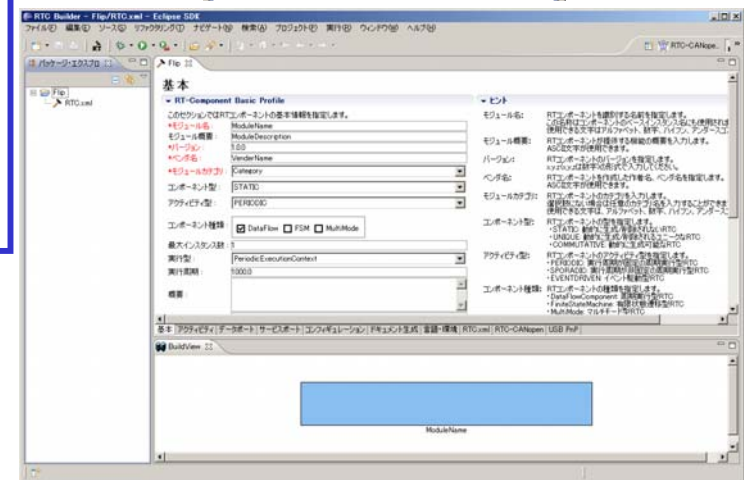
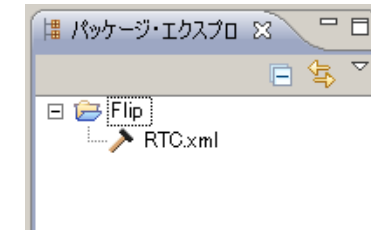
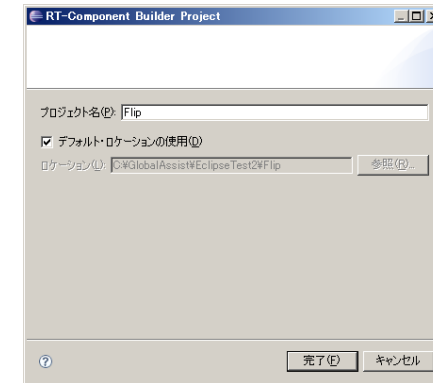


- ※メニューから「ファイル」-「新規」-「プロジェクト」を選択  
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し、「次へ」
- ※メニューから「ファイル」-「Open New Builder Editor」を選択

- ※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

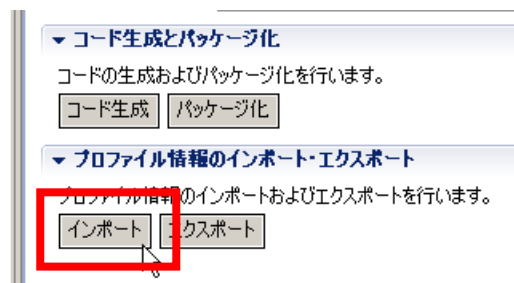
プロジェクト名: Flip

## ② 「プロジェクト名」欄に入力し、「終了」

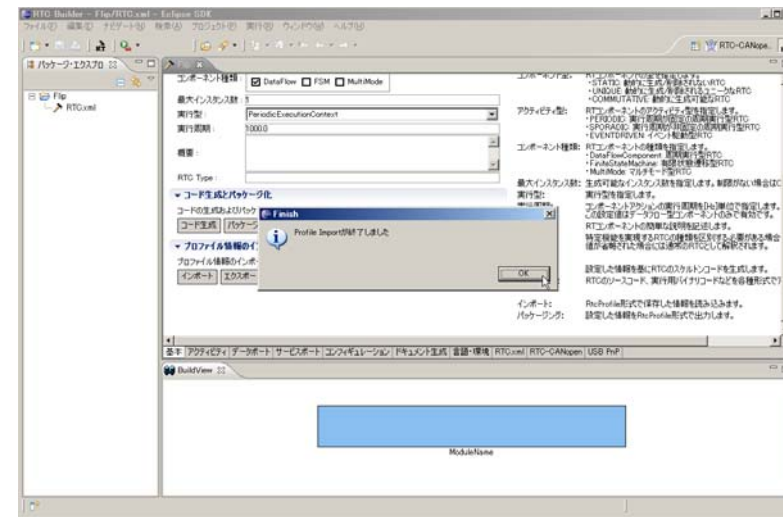


# プロファイル インポート

①「基本」タブ下部の「インポート」ボタンをクリック



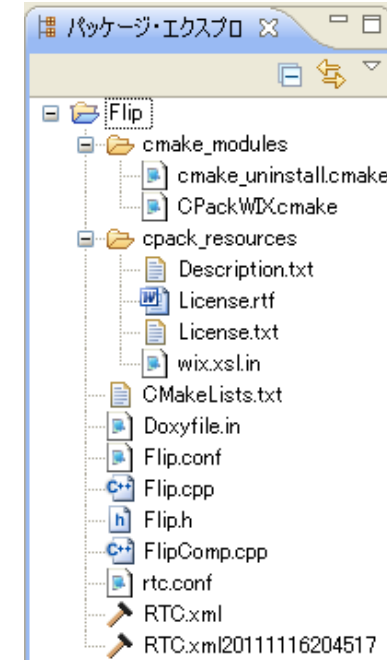
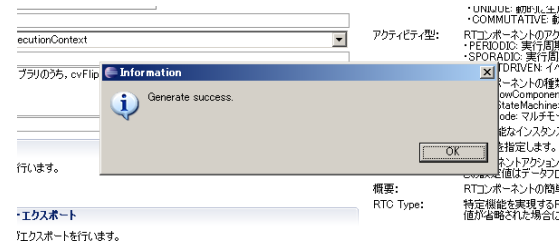
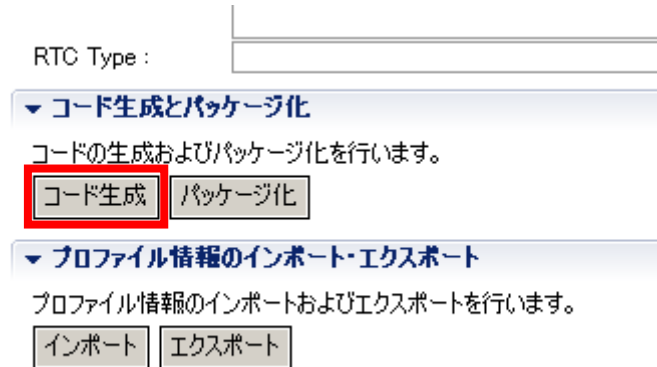
②【インポート】画面にて対象ファイルを選択



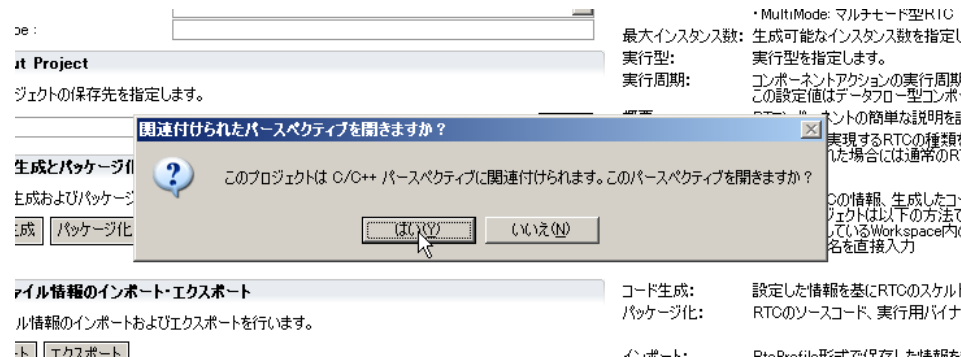
- 作成済みのRTコンポーネント情報を再利用
  - 「エクスポート」機能を利用して出力したファイルの読み込みが可能
  - コード生成時に作成されるRtcProfileの情報を読み込み可能
  - XML形式, YAML形式での入出力が可能

# 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

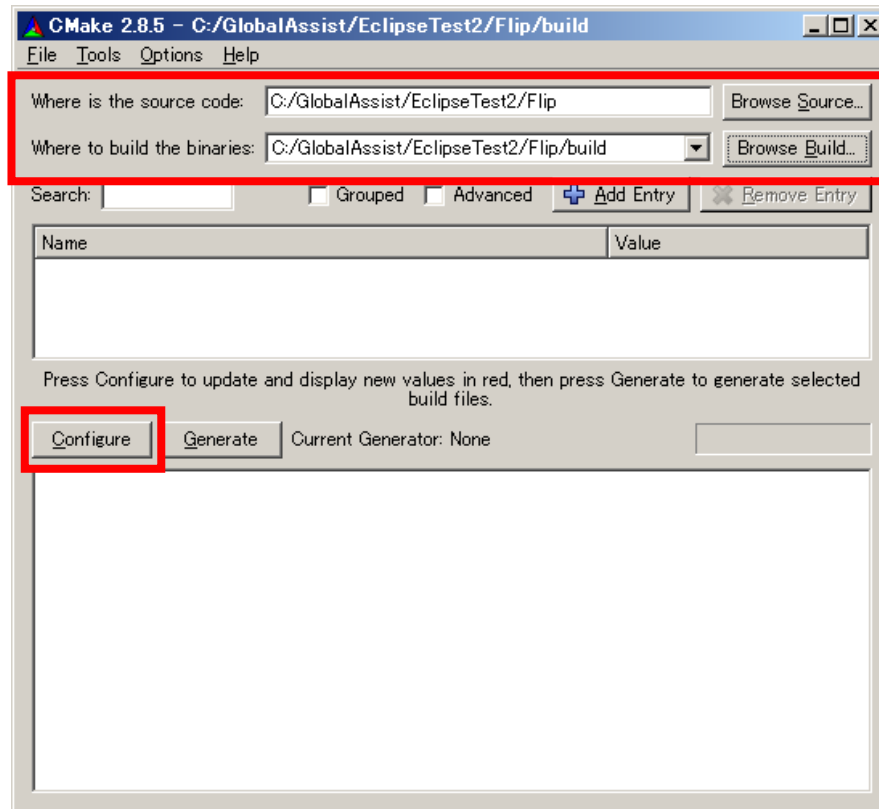
Java版RTC → JDT

(デフォルトインストール済み)

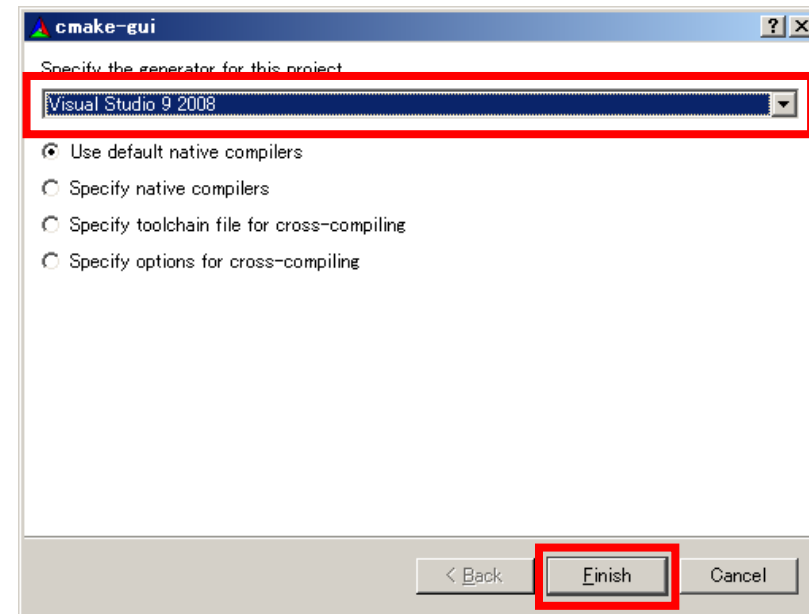
Python版 → PyDev

# 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

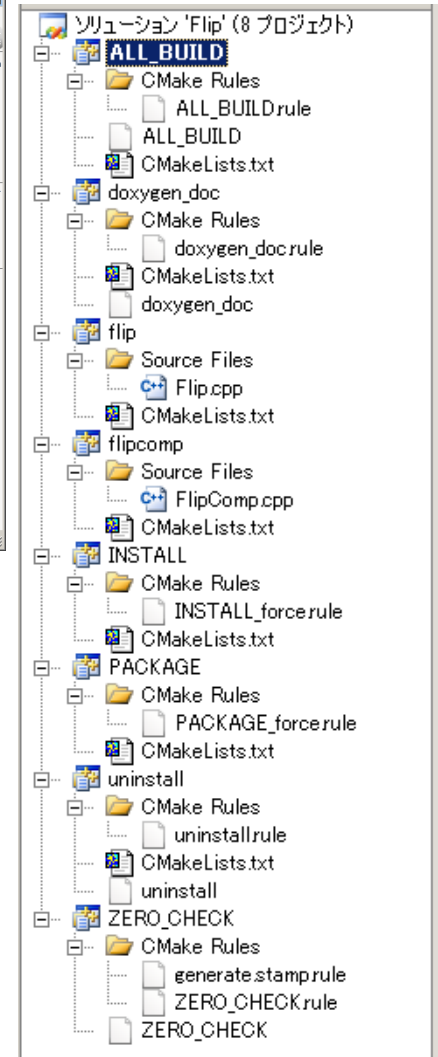
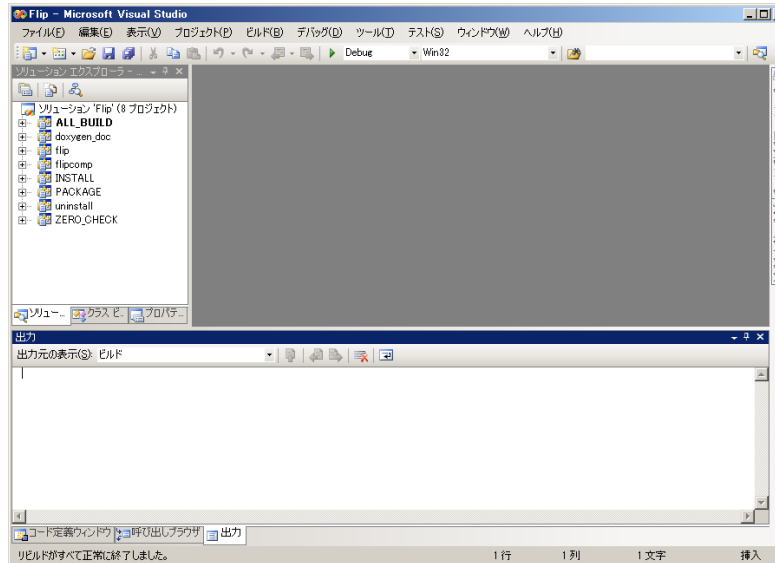
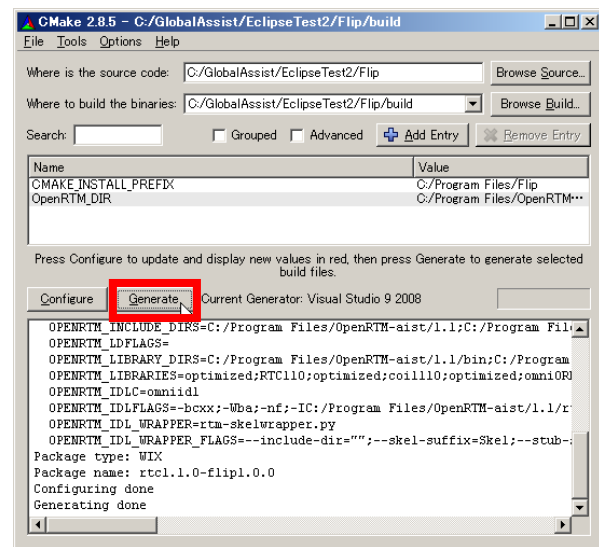


- ❌binaryには, sourceとは別のディレクトリを指定する事を推奨
- ❌日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

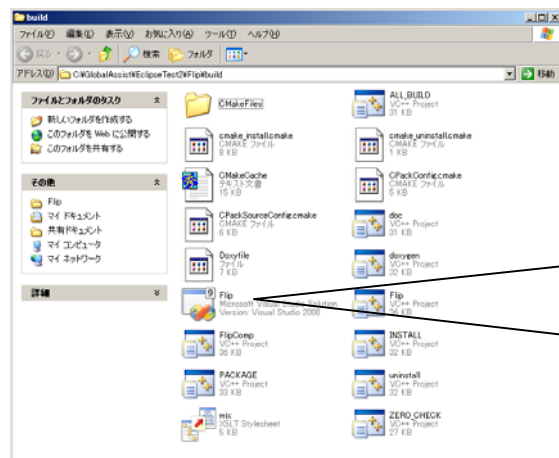


# コンパイル(Windows, CMake利用)

## ③ 正常終了後、「Generate」を実行



## 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



# RTCプロフィールエディタ



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

## ■ RTコンポーネントの名称など, 基本的な情報を設定

The screenshot shows the configuration window for an RT component named 'Flip'. The 'Basic Profile' section is expanded, showing various fields. A red box highlights the following values:

- 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

アクティビティ

▼ 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:   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一度だけ呼びれます。  
onStartup: ExecutionContextが実行を開始するとき一度だけ呼びれます。  
onShutdown: ExecutionContextが実行を停止するとき一度だけ呼びれます。  
onActivated: 非アクティブ状態からアクティブ化されたとき一度だけ呼びれます。  
onDeactivated: アクティブ状態から非アクティブ化されたとき一度だけ呼びれます。  
onAborting: ERROR状態に入る前に一度だけ呼びれます。  
onError: ERROR状態にい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一度だけ呼びれます。  
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。

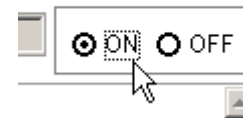
動作概要: アクティビティの概要説明を記述します。  
事前条件: アクティビティを実行する前に成立すべき事前条件を記述します。  
事後条件: アクティビティを実行した後成立すべき事後条件を記述します。

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 PrP | 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RT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)	*ポート名 (OutPort)
originalImage	flippedImage

**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  
概要説明: キャプチャされた画像データ  
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

**ヒント**

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

InPort: RTコンポーネントに他のRTコンポーネント

OutPort: RTコンポーネントから他のRTコンポーネント

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内で一意ASCII文字が使用できません

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で使用することができます

変数名: データポートに関連付けられた変数の名称は言語に

ポートの場所: RTSystemEditorなどのこのプロファイルはオプション

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レベルの情報を記述す

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  
RTC::BumperGeometry  
RTC::CameraImage  
表示位置: RTC::CameraInfo  
RTC::Carlike

Documentation

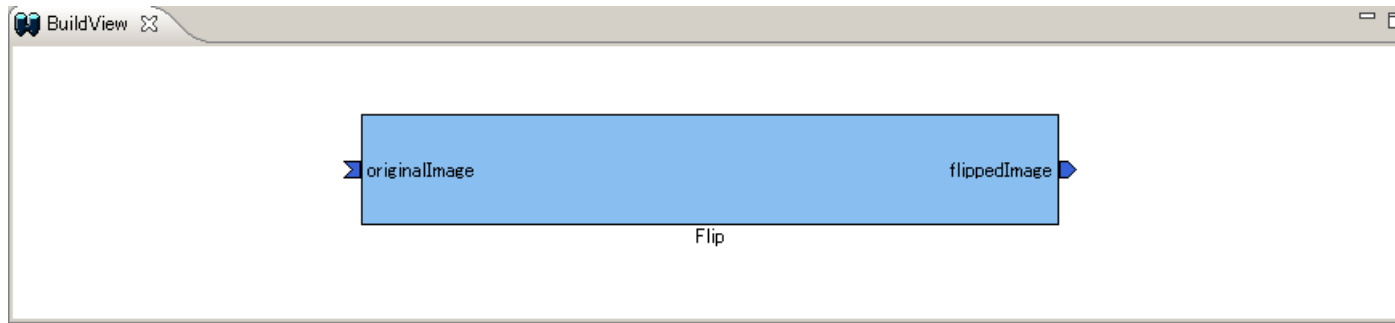
※データ型は、型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

※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

# 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

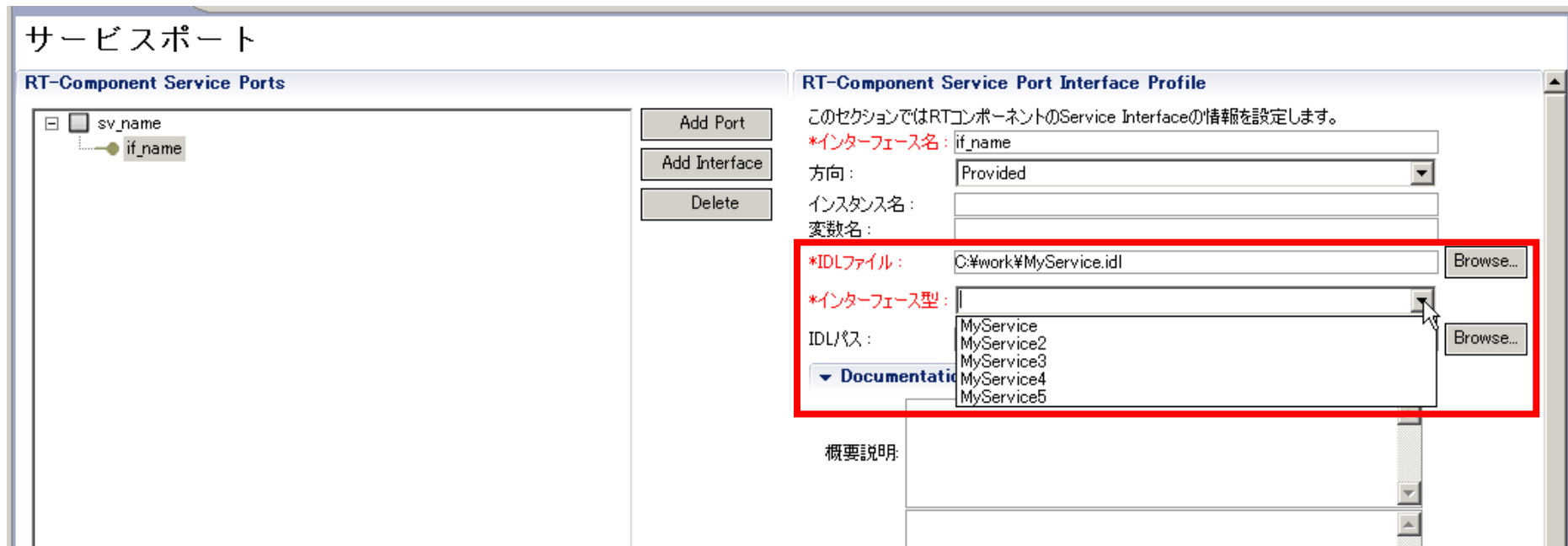
ポート名: **flippedImage**

データ型: **RTC::CameraImage**

変数名: **flippedImage**

表示位置: **right**

## ■ 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

## ■ 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

ヒント  
Config. Param.: RTコン  
再利  
パラメ  
パラメータ名: コンフ  
パラメ  
名前  
データ型: コンフ  
基本!  
デフォルト値: コンフ  
RTコ  
解釈  
変数名: コンフ  
実際  
単位: コンフ  
制約条件: コンフ  
指定  
・100  
・範囲  
・列挙  
・配列  
・ハッシュ  
Widget: コンフ  
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)  $0 \leq x \leq 100$
- 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	CPU

Add  
Delete

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**

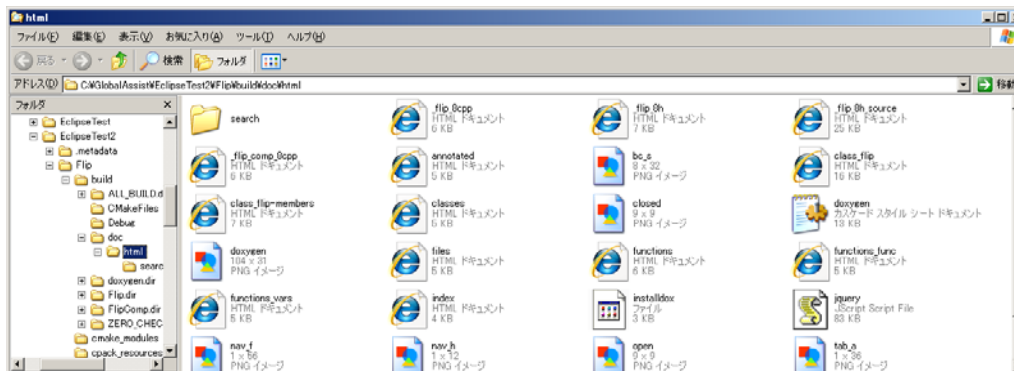
# RTCBuilder補足説明

---



# ドキュメント作成 (Windows, CMake利用)

※binaryにて指定したディレクトリ以下のdoc/html/doxygen/html以下にドキュメント



## ■ 生成されたドキュメントの例

flip 1.0.0

メインページ		クラス	ファイル
構成	構成索引	構成メンバ	
<b>クラス Flip</b>			
Flip image component. [詳細]			
#include <Flip.h>			
すべてのメンバー一覧			
<b>Public メソッド</b>			
	Flip (RTC::Manager *manager)	constructor	
	~Flip ()	destructor	
virtual RTC::ReturnCode_t	onInitialize ()		
virtual RTC::ReturnCode_t	onActivated (RTC::UniqueId ec_id)		
virtual RTC::ReturnCode_t	onDeactivated (RTC::UniqueId ec_id)		
virtual RTC::ReturnCode_t	onExecute (RTC::UniqueId ec_id)		
<b>Protected 変数</b>			
int	m_flipMode		
CameraImage	m_originalImage		
InPort< CameraImage >	m_originalImageIn		
CameraImage	m_flippedImage		
OutPort< CameraImage >	m_flippedImageOut		
<b>説明</b>			
Flip image component.			
InPortからの入力画像を反転しOutPortから出力するコンポーネント。反転の対象軸は、RTCのコンフィギュレーション機能を利用してflipModeとしめる前のパラメータで指定します。flipModeは、反転したい方向に応じて下記のように指定してください。			
<ul style="list-style-type: none"> <li>• 上下反転したい場合、0</li> <li>• 左右反転したい場合、1</li> <li>• 上下左右反転したい場合、-1</li> </ul>			
作成するRTCの入出力仕様は以下のとおりです。			

**関数**

RTC::ReturnCode\_t Flip::onActivated ( RTC::UniqueId ec\_id ) [virtual]

データ領域の確保  
・イメージ用メモリの初期化  
・outPortの画像フェイスの初期化

RTC::ReturnCode\_t Flip::onDeactivated ( RTC::UniqueId ec\_id ) [virtual]

データ領域の解放  
・イメージ用メモリの解放

RTC::ReturnCode\_t Flip::onExecute ( RTC::UniqueId ec\_id ) [virtual]

Flip処理  
・新規データのチェック  
・InPortの画像データと内部バッファコピー  
・内部バッファの画像データを反転  
・反転した画像データをOutPortにコピー

RTC::ReturnCode\_t Flip::onInitialize ( ) [virtual]

コンポーネント自身の各種初期化処理

**変数**

int Flip::m\_flipMode [protected]

画像の反転方法を指定するパラメータ

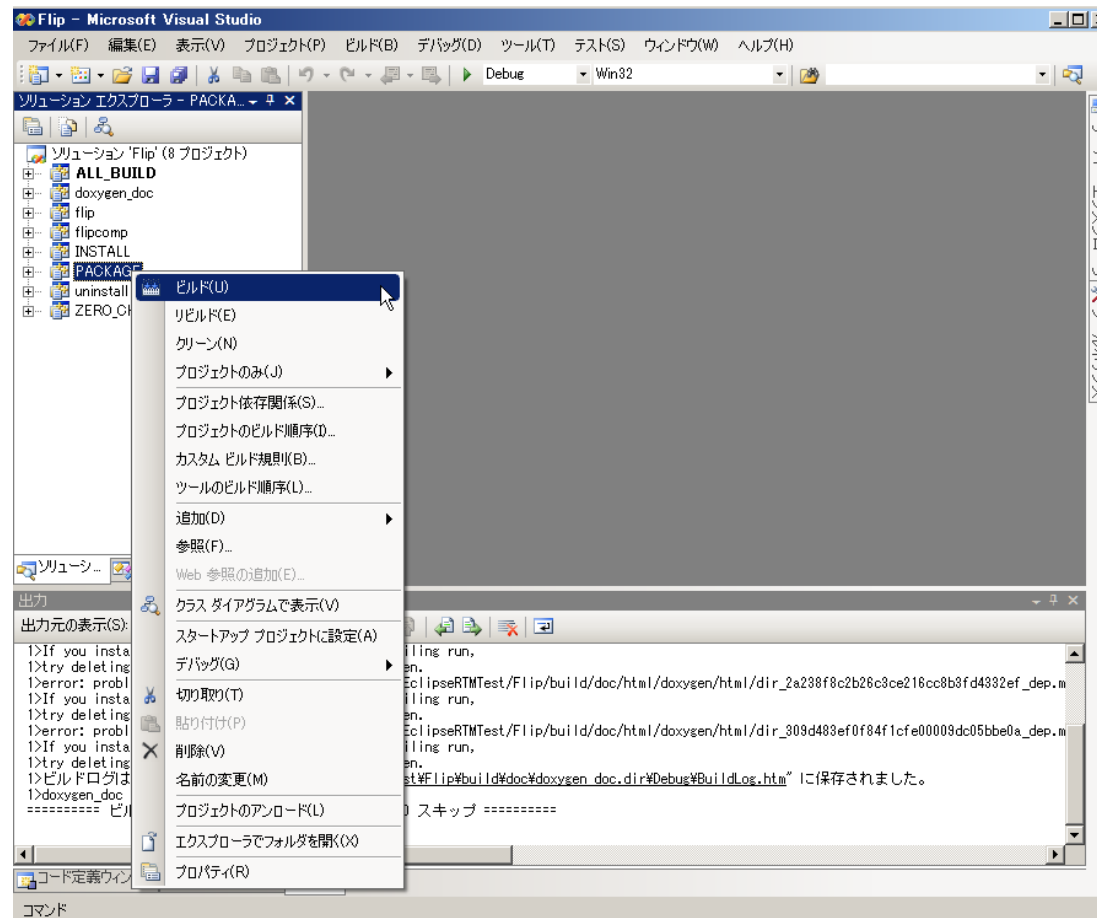
- Name: flipMode flipMode
- DefaultValue: 0
- Unit: なし

flip 1.0.0

メインページ	クラス	ファイル
ファイル一覧		
<b>C:/GlobalAssist/EclipseTest2/Flip/Flip.h</b>		
[説明を見る。]		
<pre> 00001 // -*- C++ -*- 00024 #ifndef FLIP_H 00025 #define FLIP_H 00026 00027 #include &lt;rtc/Manager.h&gt; 00028 #include &lt;rtc/DataFlowComponentBase.h&gt; 00029 #include &lt;rtc/CorbaPort.h&gt; 00030 #include &lt;rtc/DataInPort.h&gt; 00031 #include &lt;rtc/DataOutPort.h&gt; 00032 #include &lt;rtc/idl/BasicDataTypesSkel.h&gt; 00033 #include &lt;rtc/idl/ExtendedDataTypesSkel.h&gt; 00034 #include &lt;rtc/idl/InterfaceDataTypesSkel.h&gt; 00035 00036 // Service implementation headers 00037 // &lt;rtc-template block="service_impl.h"&gt; 00038 00039 // &lt;/rtc-template&gt; 00040 00041 // Service Consumer stub headers 00042 // &lt;rtc-template block="consumer_stub.h"&gt; 00043 00044 // &lt;/rtc-template&gt; 00045 00046 using namespace RTC; 00047 00078 class Flip 00079 : public RTC::DataFlowComponentBase 00080 { 00081 public: 00086 Flip (RTC::Manager* manager); 00087 00088 ~Flip (); 00092 00093 // &lt;rtc-template block="public_attribute"&gt; 00094 00095 // &lt;/rtc-template&gt; 00096 00097 // &lt;rtc-template block="public_constructor"&gt; </pre>		

# 配布用パッケージ作成(Windows,CMake利用)

## ■ ソリューション中の「PACKAGE」をビルド



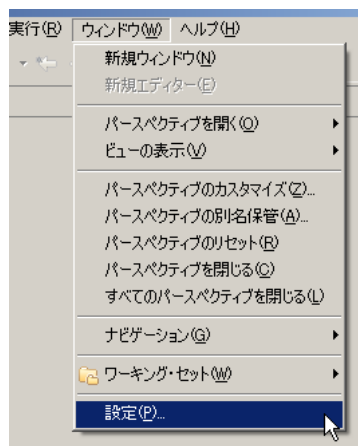
- binaryにて指定したディレクトリ直下にmsi形式のインストールパッケージを生成
  - コンポーネントのインストール先  
C:¥Program Files¥OpenRTM-aist¥1.1¥components¥<言語>/<パッケージ名>

# 各種設定

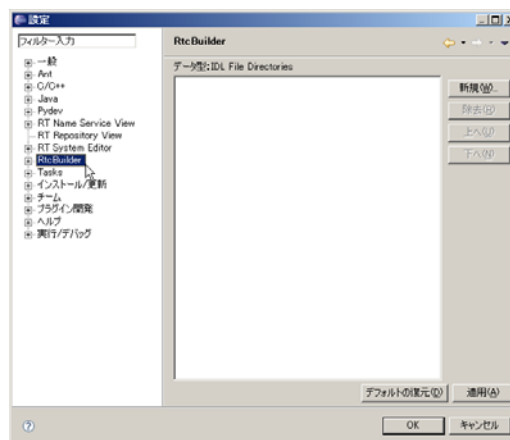
## ■ DataPortにて利用するデータ型の指定

→データ型を定義したIDLファイルが格納されているディレクトリを指定

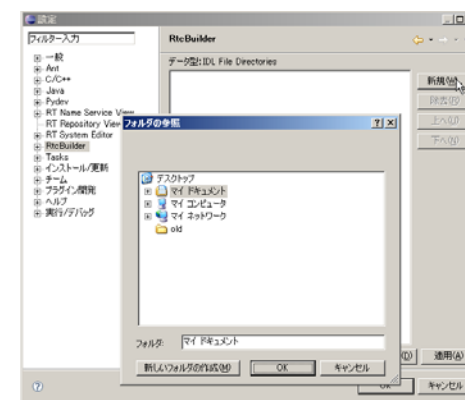
①メニューから「ウィンドウ」-「設定」



②「RtcBuilder」を選択



③「新規」ボタンにて表示されるディレクトリ選択ダイアログにて場所を指定



※独自に定義したデータ型を使用する場合のみ必要な設定

OpenRTM-aistにて標準で用意されている型のみを使用する場合には設定不要

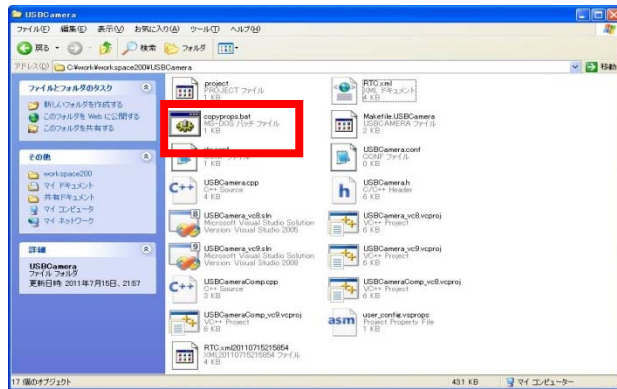
・標準型の定義内容格納位置：[RTM\_Root]rtm/idl

→BasicDataType.idl, ExtendedDataTypes.idlなど

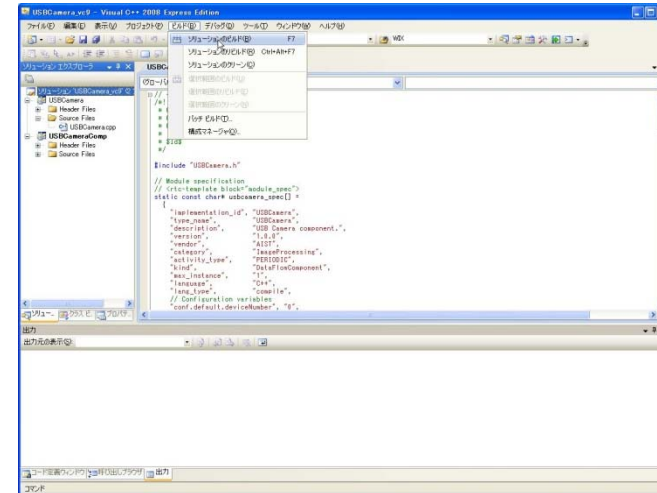
→デフォルト設定では，[RTM\_Root]=C:/Program Files/OpenRTM-aist/1.1/

# コンパイル・実行(Windows)

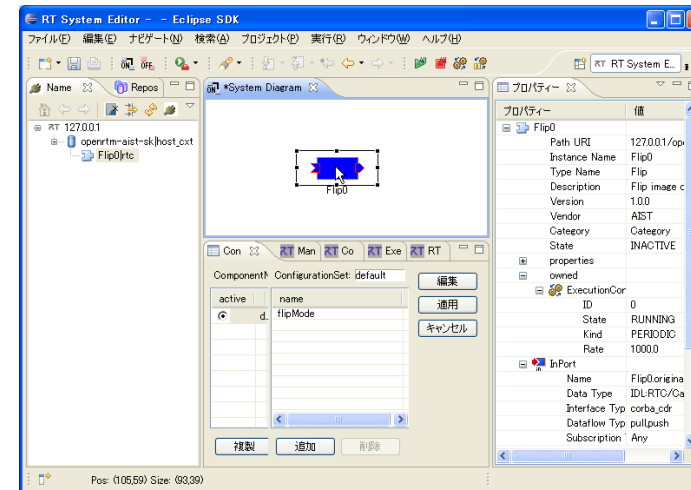
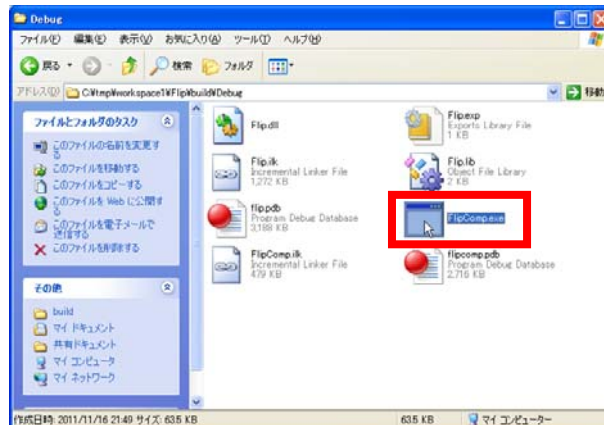
①コード生成先ディレクトリ内の「copyprops.bat」をダブルクリックして、設定ファイルをコピー



②VisualStudioを用いたビルド

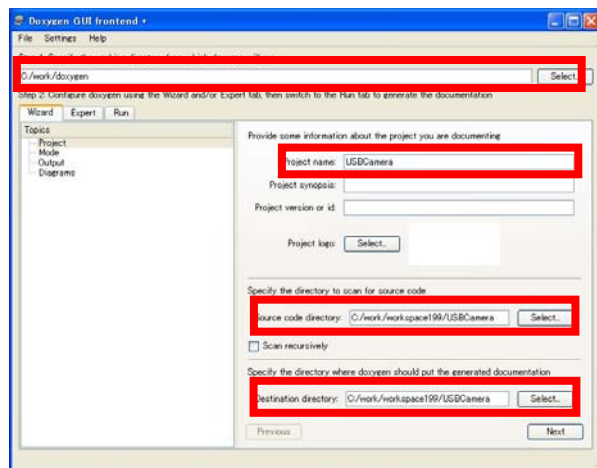


③FlipComp¥¥Debug内のFlipComp.exeを起動

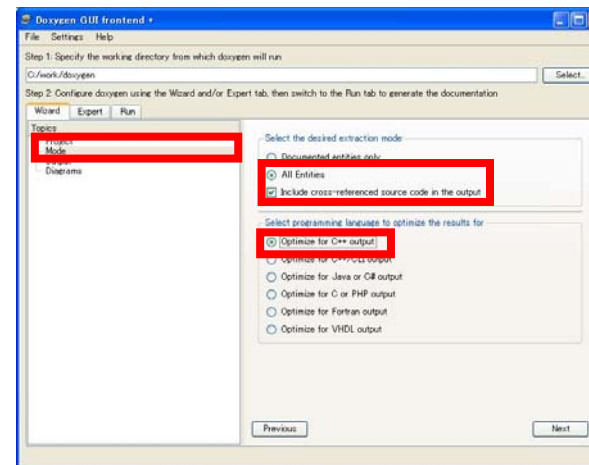


# ドキュメント作成(Windows)

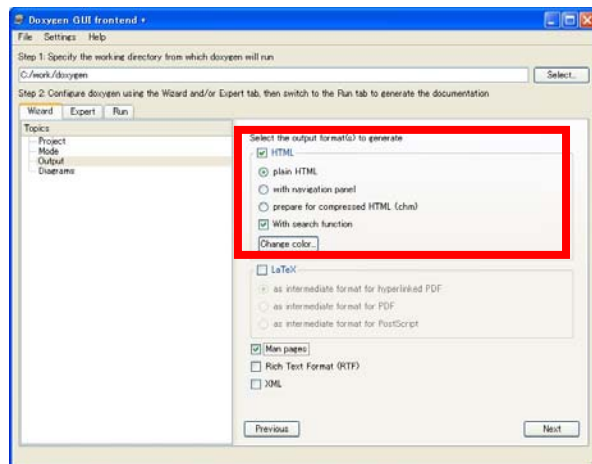
- ① Doxygen用GUIツールを起動  
作業用ディレクトリ,ソース格納場所,  
生成ファイル出力先,プロジェクト名を指定



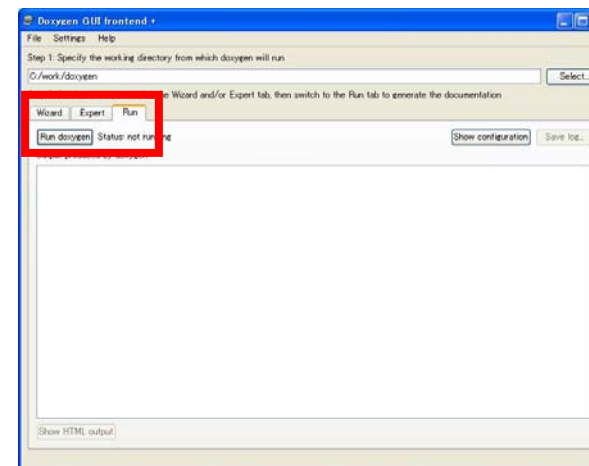
- ② 「Mode」セクションにて,  
出力内容,使用言語を指定



- ③ 「Output」セクションにて, html出力を指定



- ③ 「Run」タブにて, 「Run doxygen」を実行





# RTSystemEditor補足説明

---



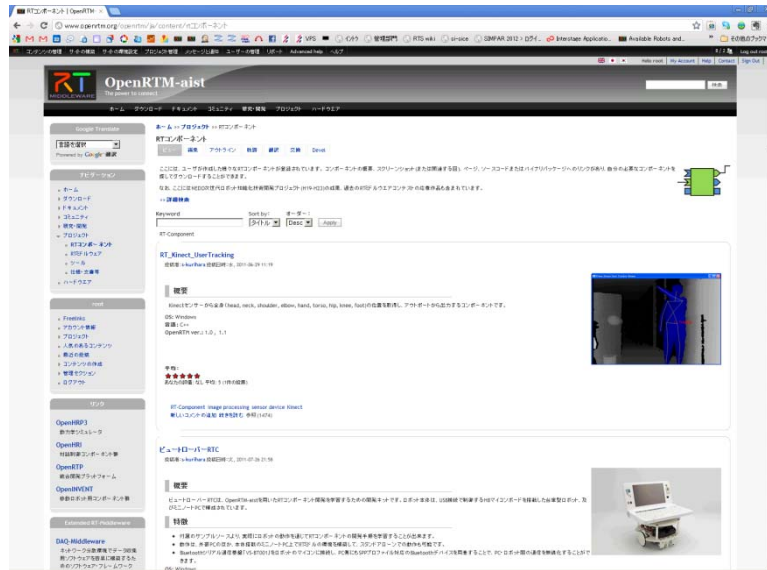
## ■ プロジェクトとは

- ユーザが作成した様々なコンポーネントやツールの公開場所
- ユーザ登録すれば、誰でも自分の成果物の紹介ページを作成可能
- 他のユーザに自分のコンポーネント等を紹介することができる

## ■ プロジェクトのカテゴリ

- RTコンポーネント: 1つのコンポーネントまたは複数のコンポーネント群などが登録されています。
- RTミドルウェア: OpenRTM-aistや他のミドルウェア、ミドルウェア拡張モジュール等が登録されています。
- ツール: 各種ツール(RTSystemEditorやrtshellを含む)ツールはこのカテゴリになります。
- 関連ドキュメント: 関連ドキュメントとは、各種インターフェースの仕様書やマニュアル等を含みます。

# プロジェクトページ



タイプ	登録数
RTコンポーネント群	638
RTミドルウェア	29
ツール	39
仕様・文書	4
ハードウェア	30

# 既存コンポーネントの再利用

## ■ プロジェクトから対象コンポーネントを取得

### ■ 「顔検出コンポーネント」

<http://www.openrtm.org/openrtm/ja/project/facedetect>

対象コンポーネントをダウンロード

OpenRTM-aist  
The power to connect

ホーム >> プロジェクト >> RTコンポーネント >> 顔検出コンポーネント

顔検出コンポーネント

投稿者: s-kurihara 投稿日時: 火, 2011-11-15 18:20

同合先(メールアドレス): [openrtm@openrtm.org](mailto:openrtm@openrtm.org)

InPortから入力された画像から顔を検出し、入力画像に検出結果を描画した画像をOutPortから出力するとともに、別のOutPortでは、顔の位置や検出した顔の数を出力するコンポーネントです。

詳細は、[こちら](#)をご覧ください。

バージョン	Downloads	日付	Links
0.1	<a href="#">Download (17.44 MB)</a>	2011-11-15	<a href="#">Notes</a>

OS: Windows  
言語: C++  
OpenRTM ver.: 1.1  
☆☆☆☆☆

# 既存コンポーネントの再利用

- ダウンロードしたファイル(FaceDetect.zip)を解凍
- 解凍したディレクトリ内の以下のファイルを実行し、システムエディタ上に配置  
\$(FaceDetect\_Root)/build/Release/FaceDetectComp.exe

The screenshot shows the RT System Editor interface. The main window displays a system diagram with components: DirectShowCam0, Flip0, FaceDetect0, Edge0, and CameraViewer0. The configuration window for FaceDetect0 is open, showing the following table:

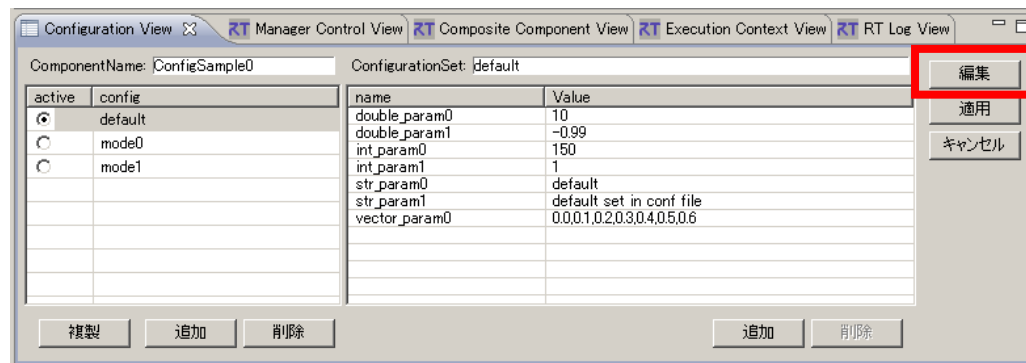
ComponentName	ConfigurationSet	name	Value
FaceDet	default	downscale	1.2
		haarcascade	../data/haarcascades/haarc...
		min_object_height	30
		min_object_width	30

On the right, the Properties window shows details for FaceDetect0, including Path URI, Instance Name, Type Name, Description, Version, Vendor, Category, State, and ExecutionCor. Below the Properties window, a small window titled 'Capture Image' shows a video feed of a person with a red bounding box around their face, indicating successful face detection.

# ネットワーク上の他のRTCとの接続

- IPアドレスの確認
  - スタートメニュー中の「全てのプログラム」-「アクセサリ」-「コマンドプロンプト」
  - コマンド「ipconfig」を実行
- 他PC上で動作するRTCとの接続
  - 隣の方のIPアドレスを聞く
  - RTSystemEditorの「ネームサーバを追加(コンセントのアイコン)」をクリックして、上記のIPアドレスを入力する
  - 隣の方のネームサーバ内の階層化にあるDirectShowCamをシステムエディタにDnDする
  - 上記でDnDしたDirectShowCamと自分のPC上で起動したCameraViewerのデータポートを接続する

## ■ RTコンポーネントのコンフィギュレーション情報の確認/編集



- ※「編集」ボタンにより、各種コントロールを用いた一括編集が可能
- ※「Apply」チェックボックスがONの場合、設定値を変更すると即座にコンポーネントに反映  
→テキストボックスからフォーカス外れる、ラジオボタンを選択する、スライダーを操作する、スピナを変更する、などのタイミング
- ※コンフィギュレーション情報を複数保持している場合、上部のタブで編集対象を切り替え

# コンフィギュレーション情報の設定方法

- rtc.conf内

[カテゴリ名]. [コンポーネント名]. config\_file: [コンフィギュレーションファイル名]

※例) example.ConfigSample.config\_file: configsample.conf

- コンフィギュレーションファイル内

- コンフィギュレーション情報

conf. [コンフィグセット名]. [コンフィグパラメータ名] : [デフォルト値]

※例) conf.mode0.int\_param0: 123

- Widget情報

conf. \_\_widget\_\_. [コンフィグパラメータ名] : [Widget名]

※例) conf.\_\_widget\_\_.str\_param0: radio

- 制約情報

conf. \_\_constraints\_\_. [コンフィグパラメータ名] : [制約情報]

※例) conf.\_\_constraints\_\_.str\_param0: (bar,foo,foo,dara)

conf. \_\_[コンフィグセット名]. [コンフィグパラメータ名] : [制約情報]

※例) conf.\_\_mode1.str\_param0: (bar2,foo2,dara2)

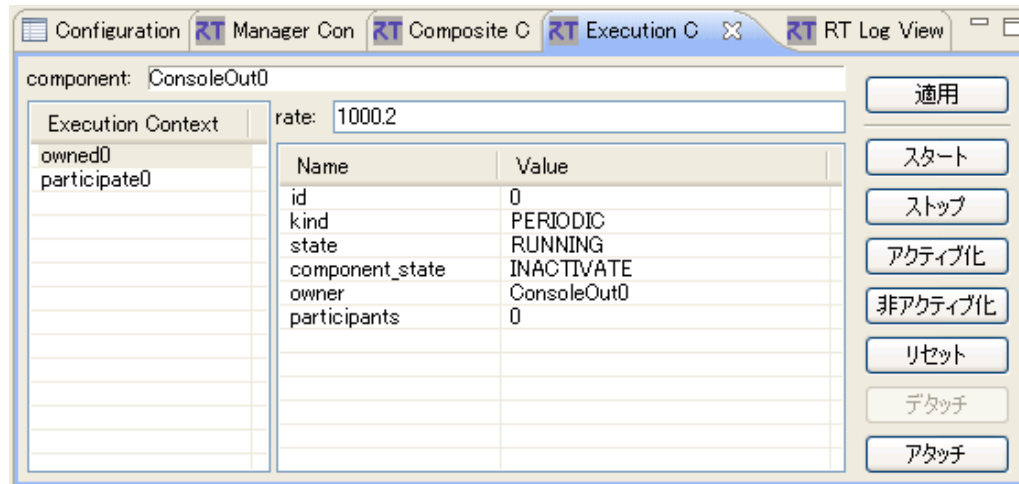
RTCの利用者が設定するのではなく、RTC開発者、RTC管理者が設定することを想定。

RTCBuilderを使用することで設定可能



# 実行コンテキストビュー

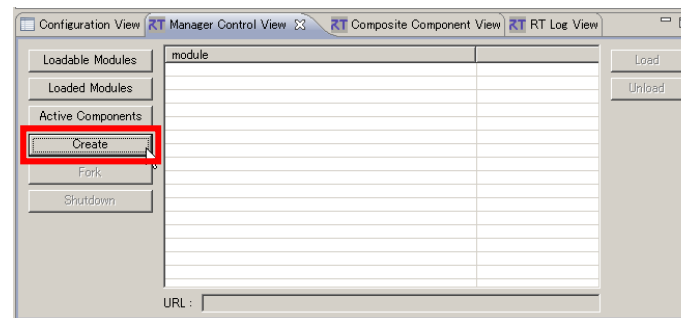
- RTコンポーネントが属する実行コンテキスト(EC)を一覧表示



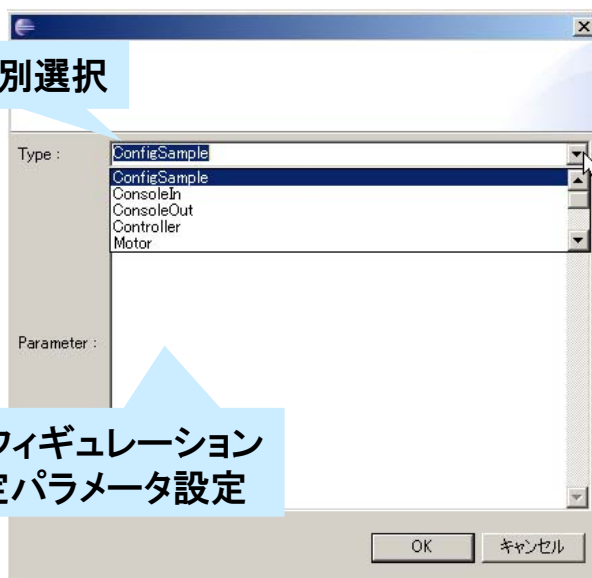
属性名	説明
id	ECのID. オンラインの場合には, context_handleを表示
kind	ECの種別(PERIODIC/EVENT_DRIVEN/OTHER)
state	ECの状態(RUNNING/STOPPING)
component state	対象RTCの状態(ACTIVE/INACTIVE/ERROR)
owner	対象ECを所有しているオーナーRTCのインスタンス名
participants	対象ECに参加中のRTCの数

※対象ECの実行周期の変更, EC自身の動作開始/終了, 新規RTCへのアタッチ, アタッチ済みRTCのデタッチも可能

## ■ RTコンポーネントの新規インスタンスの生成

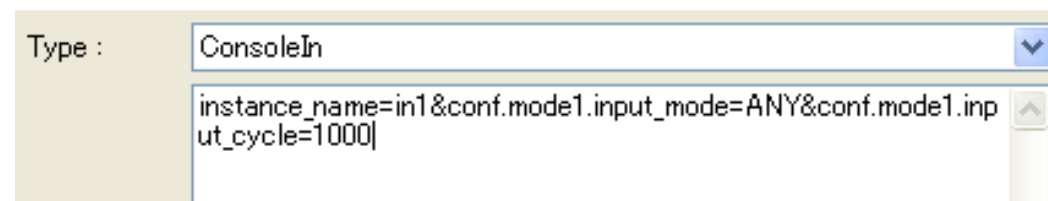


RTC種別選択

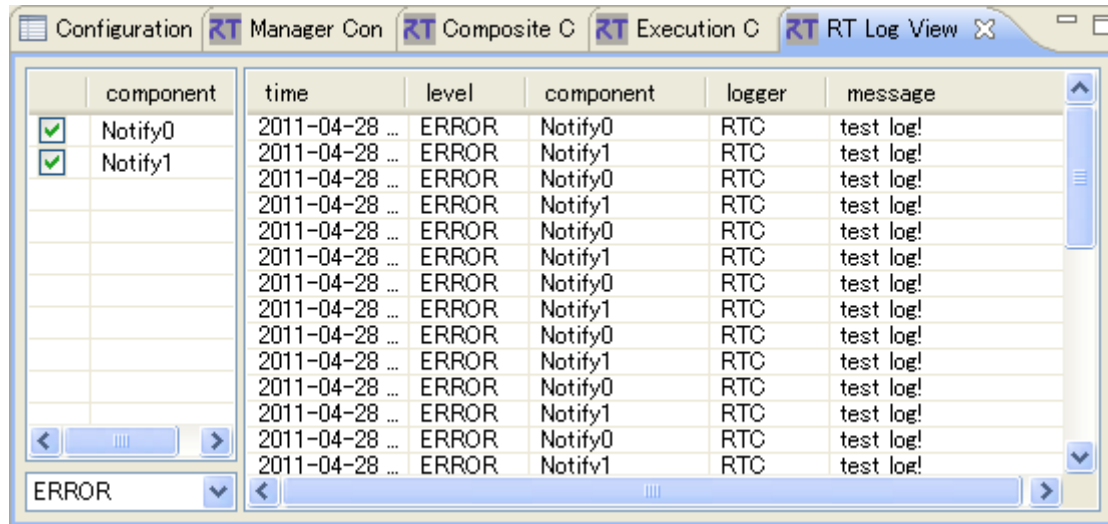


コンフィギュレーション  
指定パラメータ設定

- コンフィギュレーション指定パラメータ
  - `conf. [ConfigSet名]. [Configパラメータ名]=[設定値]`の形式にてConfigurationSetの値も設定可能

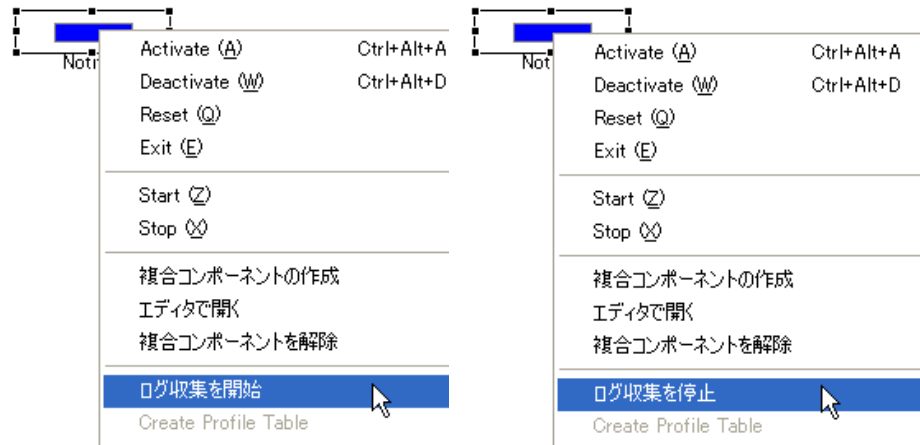


## ■ 選択したRTCから収集したログ情報を一覧表示

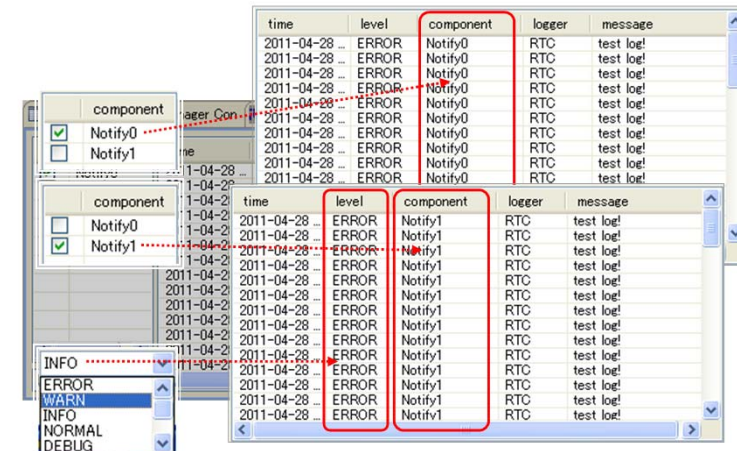


※近日機能追加予定

## ● ログ収集の開始/停止



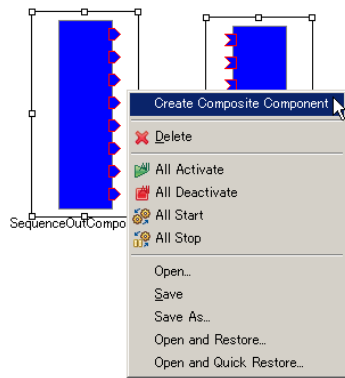
## ● ログ情報のフィルタリング



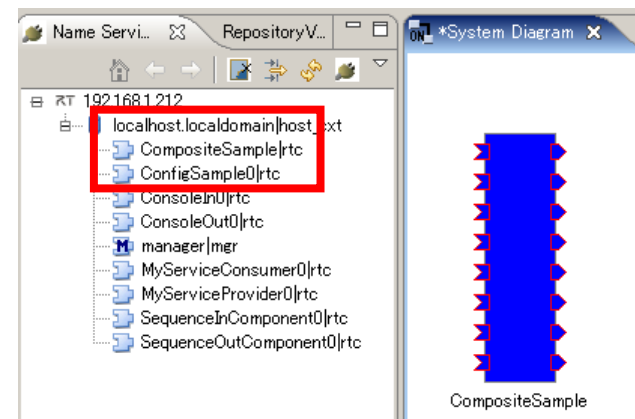
# 複合コンポーネント

- 複数のRTCをまとめて、1つのRTCとして扱うための仕組み
- 複合コンポーネントの作成方法

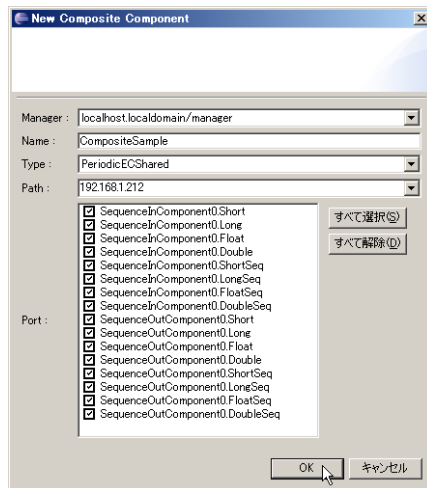
① 複数RTCを選択している状態で右クリック



③ 複合コンポーネントを生成



② 複合コンポーネントのプロパティを設定



項目	設定内容
Manager	複合コンポーネントを制御するマネージャを選択
Name	複合コンポーネントのインスタンス名を入力
Type	複合コンポーネントの型を選択
Path	複合コンポーネントのパスを入力
Port	外部に公開するポートを選択

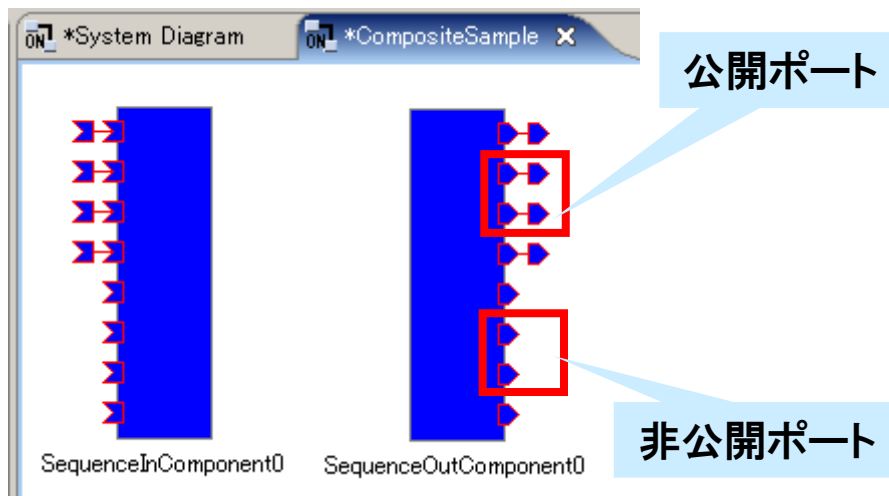
※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

## ■ 複合コンポーネントのタイプについて

タイプ名	説明
PeriodicECShared	実行主体であるExecutionContextのみを共有. 各子コンポーネントはそれぞれの状態を持つ
PeriodicStateShared	実行主体であるExecutionContextと状態を共有
Grouping	便宜的にツール上のみでグループ化

## ■ 複合コンポーネントエディタ

- 複合コンポーネントをダブルクリックすることで表示



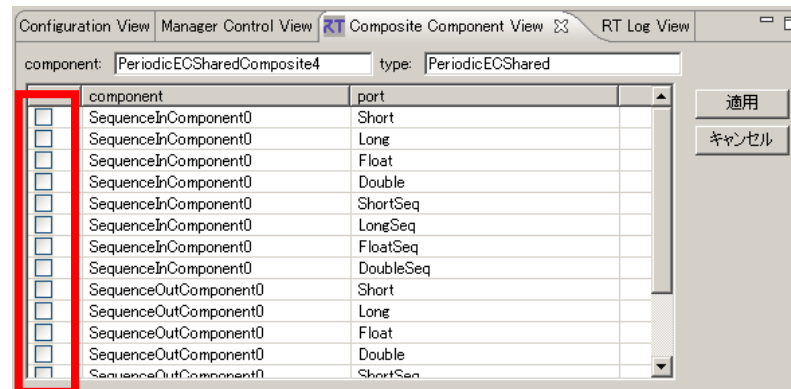
- ※エディタ内に別RTCをDnDすることで、子コンポーネントの追加が可能  
→追加したRTCのポートは全て非公開に設定
- ※エディタ内のRTCを削除することで、子コンポーネントの削除が可能  
→削除されたRTCは、親エディタに表示

# 複合コンポーネント

## ■ 公開ポートの設定

### ● 複合コンポーネントビュー

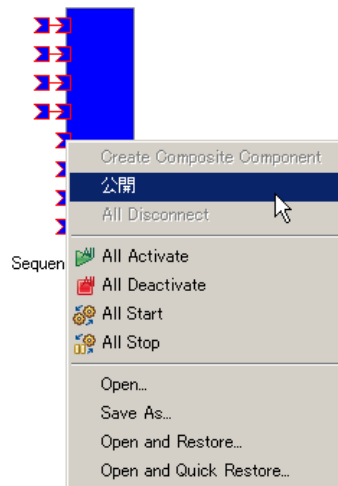
ポート公開情報



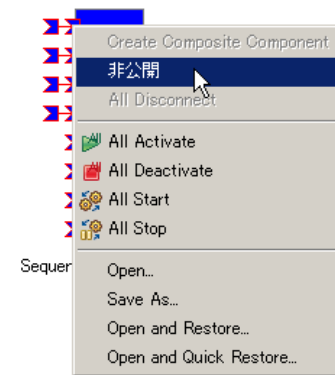
※ポート公開情報を変更し、「適用」をクリック

### ● 複合コンポーネントエディタ

※非公開ポートを「公開」



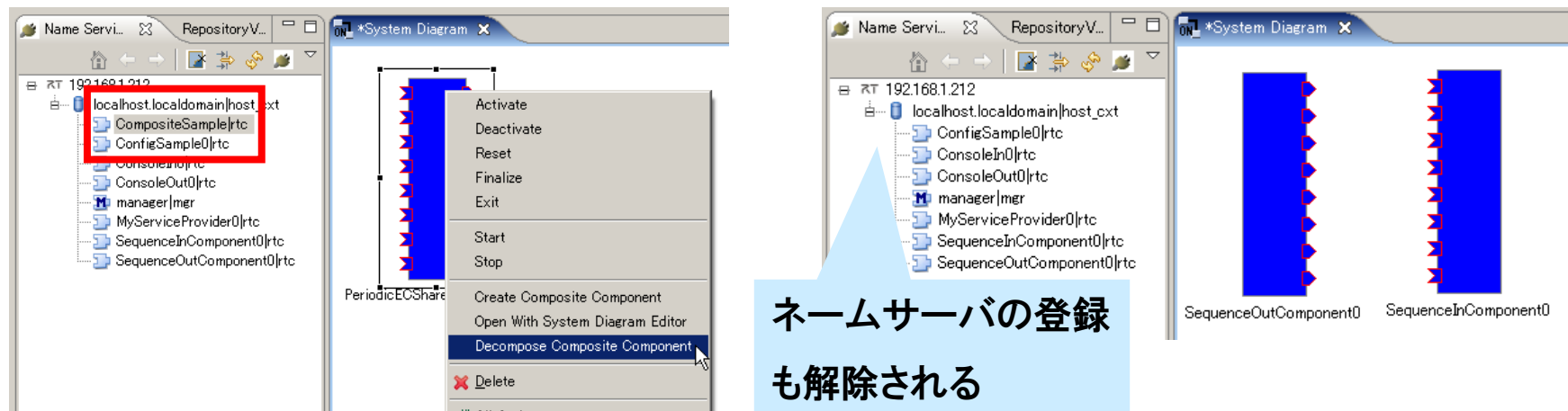
※公開ポートを「非公開」



外部コンポーネントと接続されているポートを「非公開」に設定することはできません

## ■ 複合コンポーネントの解除

- ① 複合RTCを右クリックし，複合コンポーネントの解除を選択
- ② 複合コンポーネントが分解され，内部のRTCが表示



※エディタ上で，(Deleteキーなどで)単純に削除した場合は，エディタから表示が消えるのみ複合コンポーネントは解除されない

# オフラインエディタ

- RTコンポーネントの仕様を用いてRTシステムを構築
  - 実際のRTコンポーネントが動作している必要はない

リポジトリビュー

オフライン・システムエディタ

プロパティビュー

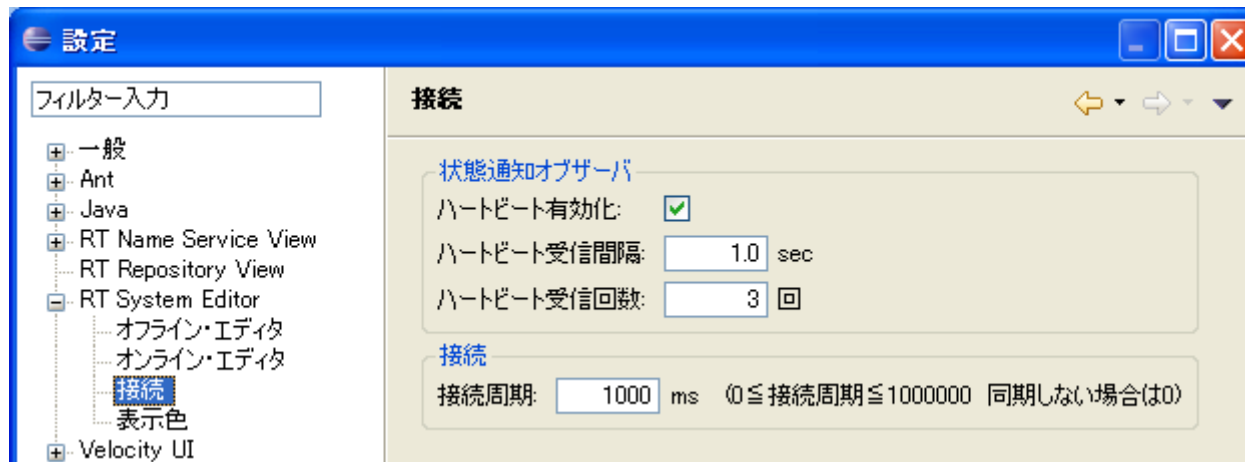
コンフィギュレーションビュー

プロパティ	値
Instance Name	ImageProcess_1
Type Name	ImageProcess
Description	Image Processing RTC
Version	1.0.0
Vendor	AIST
Category	Sample
InPort	
Name	Din
Data Type	RTC-TimedOctetSeq
OutPort	
Name	Dout
Data Type	RTC-TimedOctetSeq
ServicePort	
Name	CapPort
PortInterface	
Interface	CameraInfo
Type Name	CameraInfo
Port Inter	REQUIRED



## ■ 接続一状態通知オブザーバ

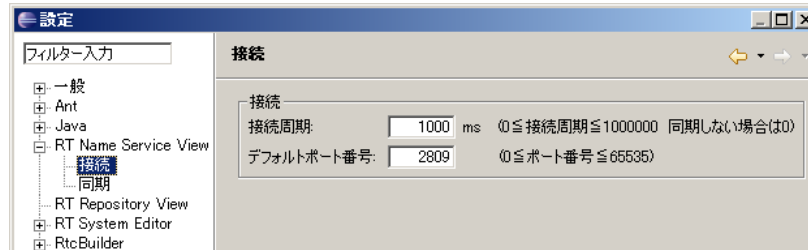
- RTCの生存確認用オブザーバに関する設定
  - RTSE側から生存確認を行うのではなく, RTC側から通知(ハートビート)を行う形
  - OpenRTM-aist-1.1以降で対応



- ハートビート有効化:ハートビートによる生存確認機能の有効化
- ハートビート受信間隔:ハートビートの受信間隔. この間隔以内にRTC側からハートビートが送られてこないと生存確認失敗と判断
- ハートビート受信回数:この回数を超えて生存確認に失敗した場合, 対象RTCに異常が発生したと判断

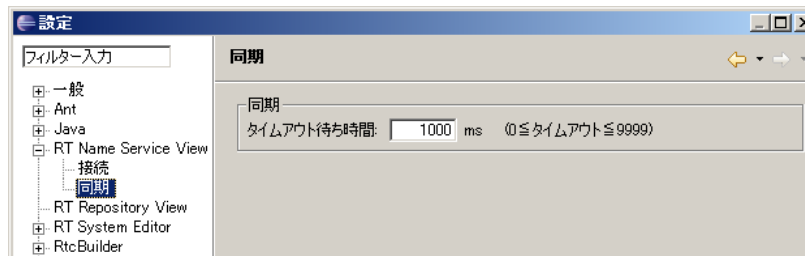
## ■ 「RT Name Service View」－「接続」【接続周期】

- ネームサービスビューが、ネームサーバに情報を問い合わせる周期



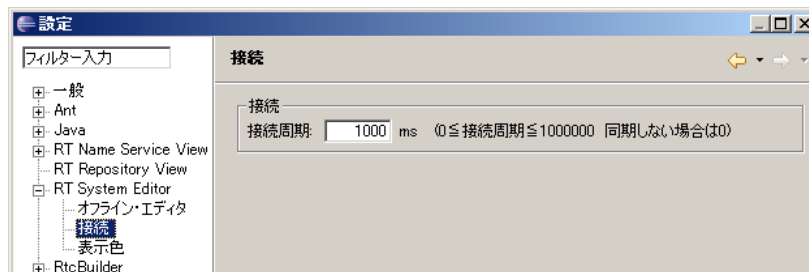
## ■ 「RT Name Service View」－「同期」【タイムアウト待ち時間】

- ネームサービスビューが、リモートオブジェクトのレスポンスを待つ時間



## ■ 「RT System Editor」－「接続」【接続周期】

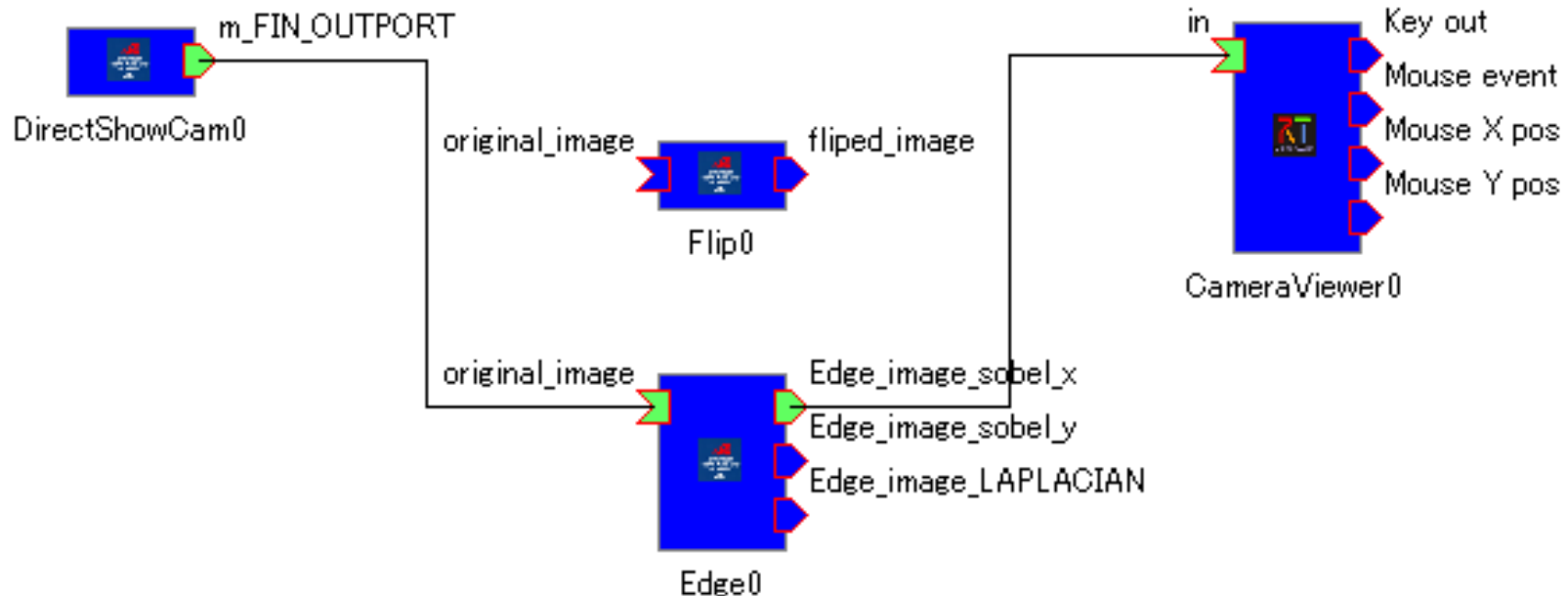
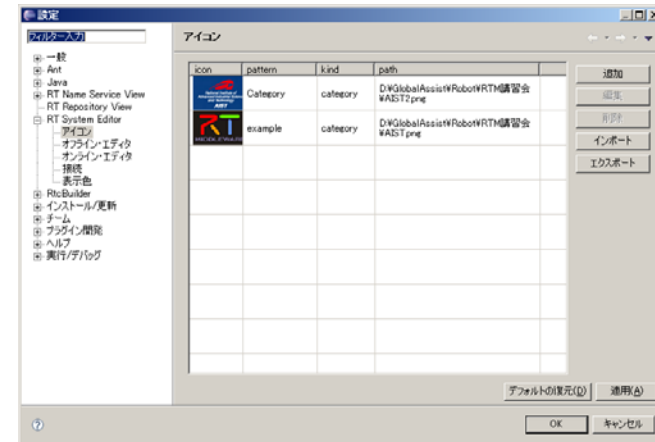
- システムエディタが、ネームサーバに情報を問い合わせる周期



**【接続周期】をゼロに設定すると  
ネームサーバとの同期を行わない**

# 設定画面

- 「RT System Editor」-「アイコン」【表示アイコン】
  - RTC内に表示するアイコンを指定可能
    - カテゴリ単位, RTC名称単位で設定が可能



# RTミドルウェア講習会

---

