

# 第2部:RTコンポーネント作成入門

名城大学

理工学部メカトロニクス工学科

大原賢一



## 第2部での目標

- RT System Editorを用いたRTCベースのシステム構築方法の習得(RTC運用時に必要な知識)
- RTC Builderを用いたRTコンポーネントのひな形作成方法の習得(RTC開発時に必要な知識)

## ■ 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

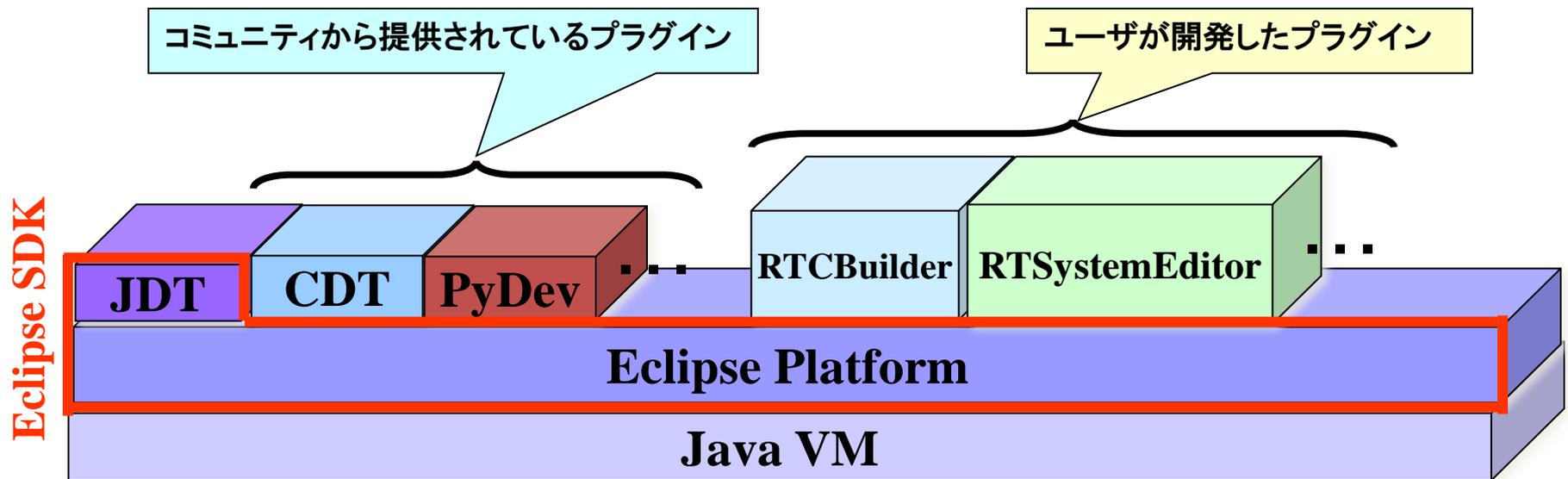
- RTCBビルダ
- RTCデバッガ
- RTシステムエディタ
- ロボット設計支援ツール
- シミュレータ
- 動作設計ツール
- シナリオ作成ツール  
など

The screenshot shows the OpenRT Platform Official Site. At the top, there is a navigation bar with tabs for Home, Software, and RTC. Below this, there is a sidebar menu with links for Downloads, Knowledge Base (Project members only), Consortium (Consortium members only), and Reuse WG (members only). The main content area features a header for 'ロボット知能ソフトウェアプラットフォーム' (Robot Intelligent Software Platform) and a 'New Information' section. The news items include:
 

- Mar 2012: JDKのバージョンライセンス変更 (JDK version license change)
- Mar 2012: OpenRTM-aist-1.0のC言語実装 (C language implementation of OpenRTM-aist-1.0)
- Oct 2011: 国際ロボット展 セミナー詳細のつか (International Robot Exhibition Seminar details)

## ■ 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



# ■ 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" and includes a "Table of contents" section with links to "全部入りパッケージ" (Full package), "バイナリ" (Binaries), "Eclipse/JDK/JRE等" (Eclipse/JDK/JRE etc.), and "過去のバージョン" (Previous versions). Below this is a table for "Eclipse-3.4.2 [Ganymede SR2]" with columns for the package name, MD5 hash, and date.

Eclipse-3.4.2 [Ganymede SR2]		
Eclipse3.4.2+RTSE+RTCB Windows用全部入り	eclipse342_rtmtools110-rc2_win32_ja.zip MD5:2e6f9fa3e370b6e7ac1f9340d36c7abf	2011.07.22

Below the table, there are instructions for installation on Ubuntu 8.04, 9.10, and 10.04, including a terminal command snippet:

```
$ 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 -f |grep xulrunner-1.9
ii xulrunner-1.9 1.9.0.8+nobinonly-0ubuntu2 xulrunner application runner
```

# システム構築支援ツール RTSystemEditorについて

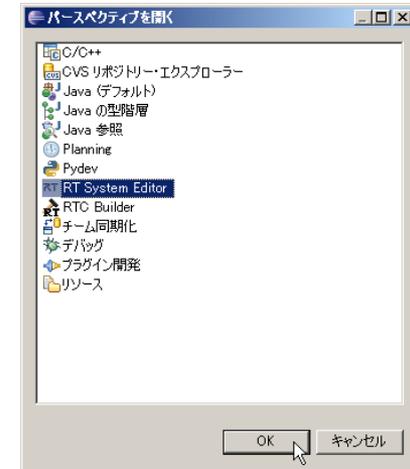


## ■ パースペクティブの切り替え

- 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

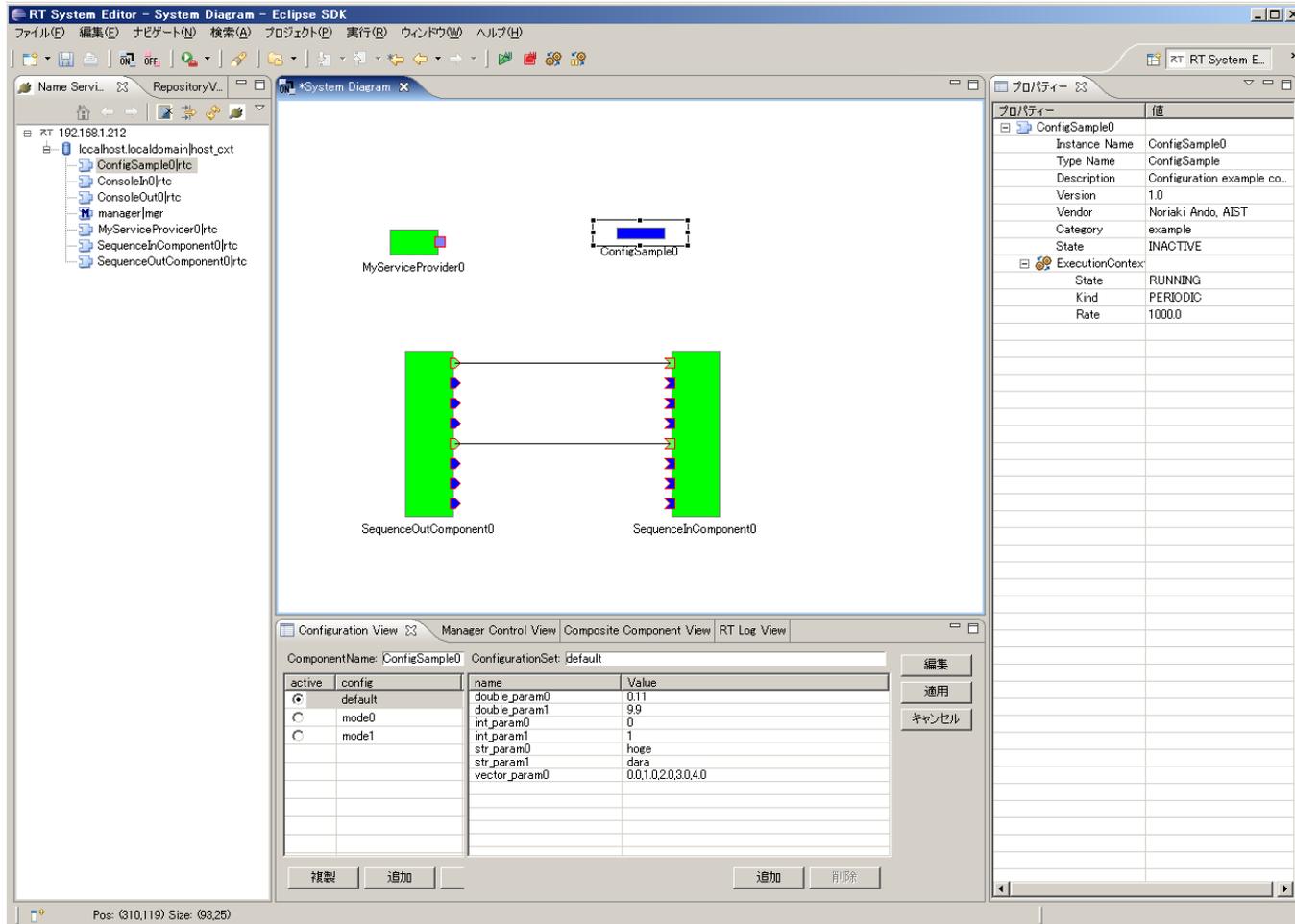


### ※ パースペクティブ

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



**システムエディタ**

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

Loadable Modules  
Loaded Modules  
Active Components  
Create  
Fork  
Shutdown

マネージャビュー

component	port
ConsoleIn0	ConsoleIn0.out
ConsoleOut0	ConsoleOut0.in

複合コンポーネントビュー

Name	Value
id	0
kind	PERIODIC
state	RUNNING

実行コンテキストビュー

component	time	level	component	logger	message
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify1	2011-04-28	ERROR	Notify1	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify1	2011-04-28	ERROR	Notify1	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify1	2011-04-28	ERROR	Notify1	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify1	2011-04-28	ERROR	Notify1	RTC	test log!
Notify0	2011-04-28	ERROR	Notify0	RTC	test log!
Notify1	2011-04-28	ERROR	Notify1	RTC	test log!

ログビュー

- 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]  
→[ConsoleInComp.exe]

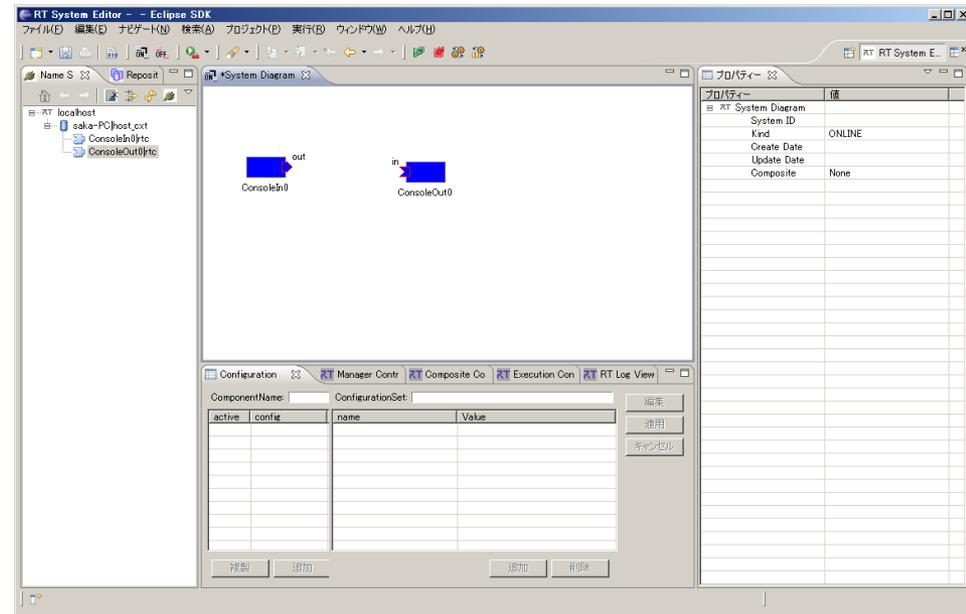
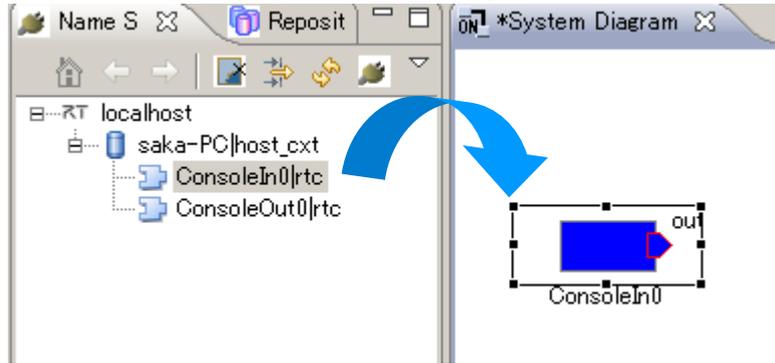
- DirectShowCamCompの起動

- [スタート]メニューから起動

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



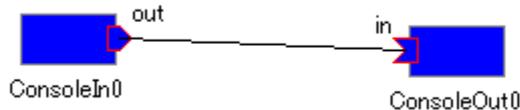
## RTコンポーネントの配置



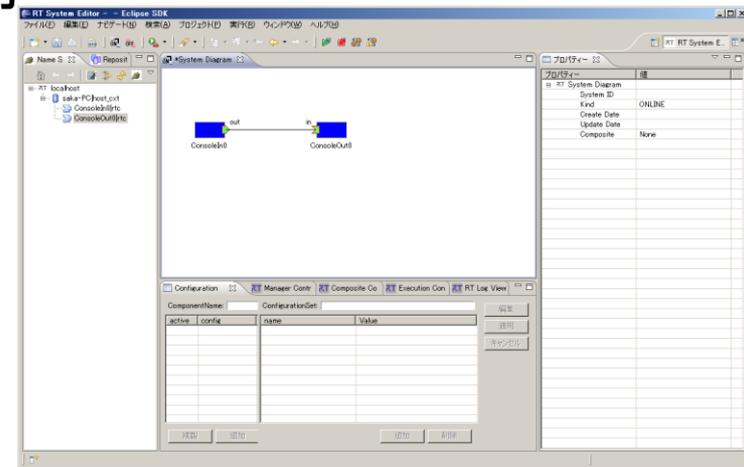
※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

## 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

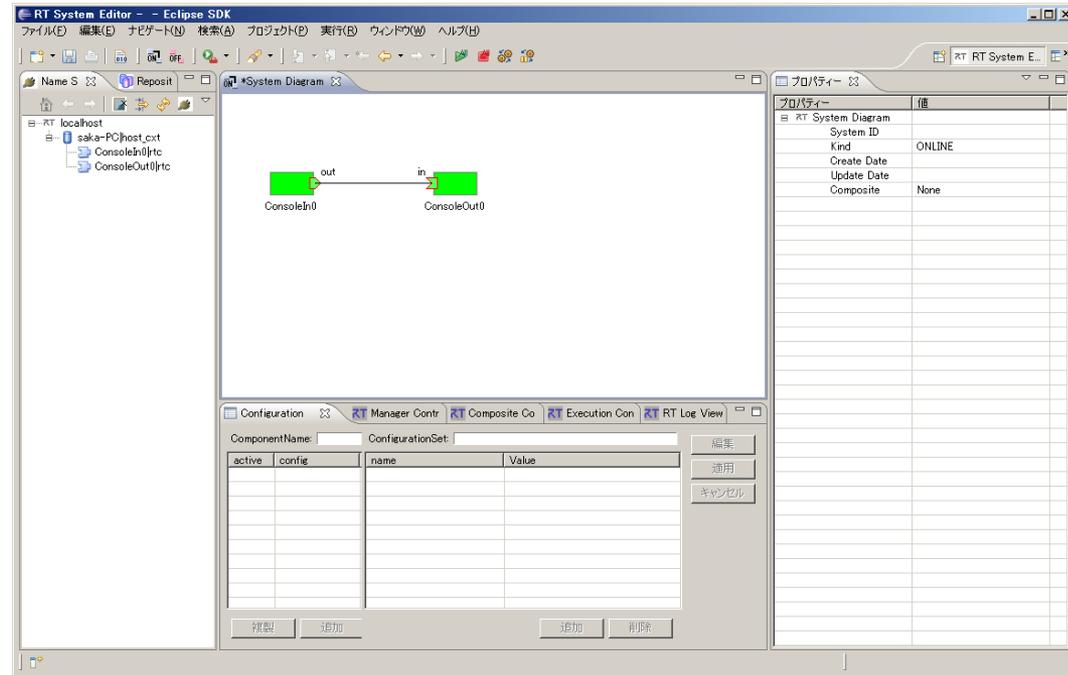
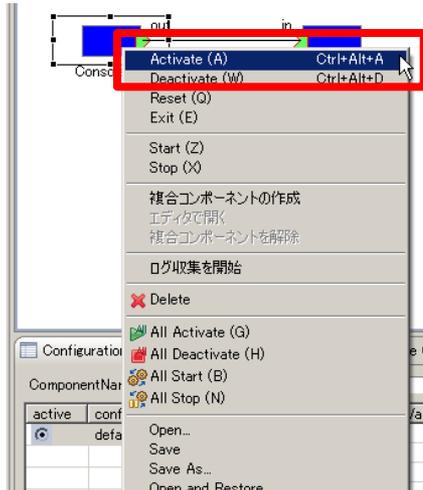


※ポートのプロパティが異なる場合など、接続不可能なポートの場合にはアイコンが変化

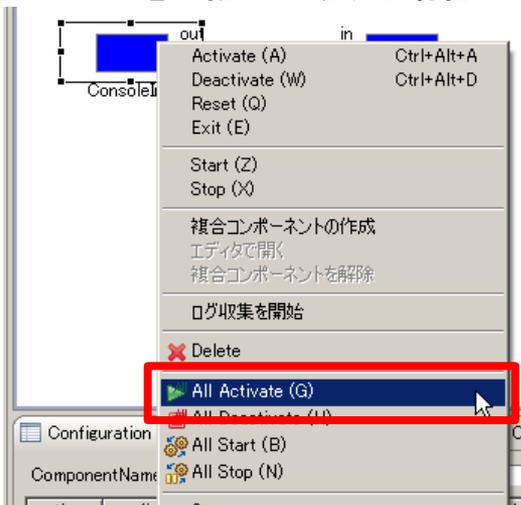


## ■ コンポーネントの起動

※各RTC単位で起動する場合

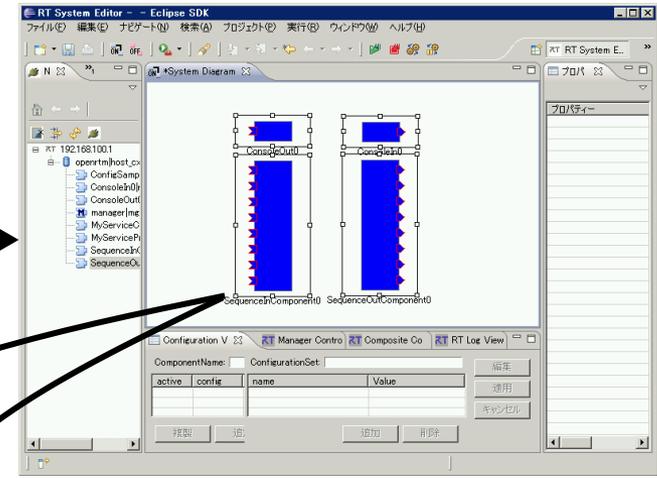
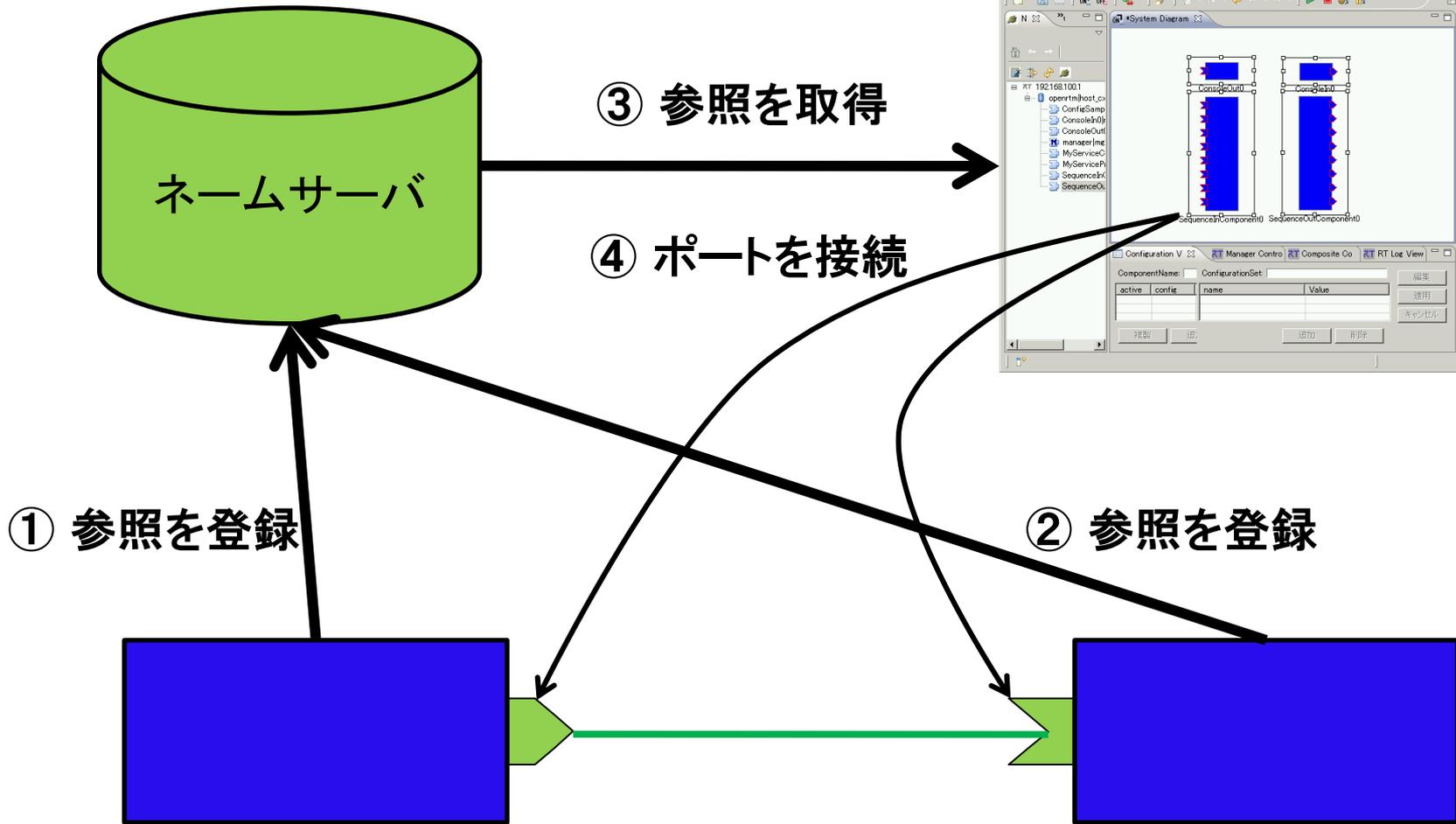


※全てのRTCを一括で起動する場合

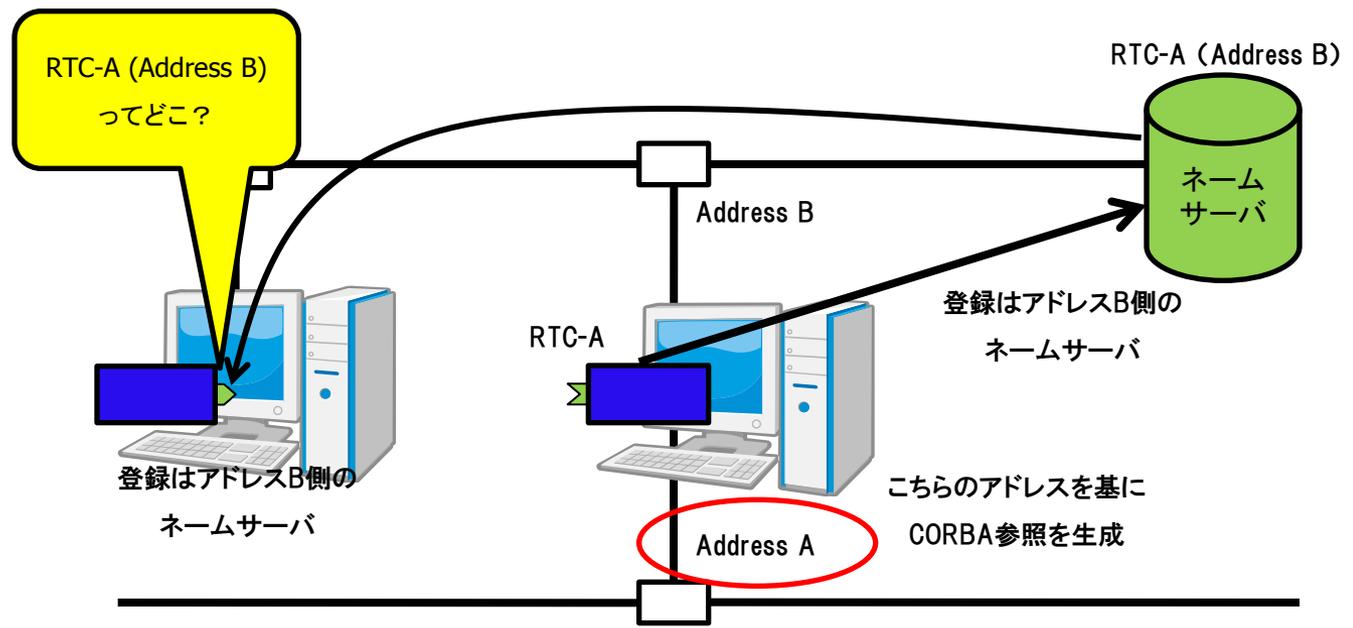


※停止はDeactivateを実行

※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



■ 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:

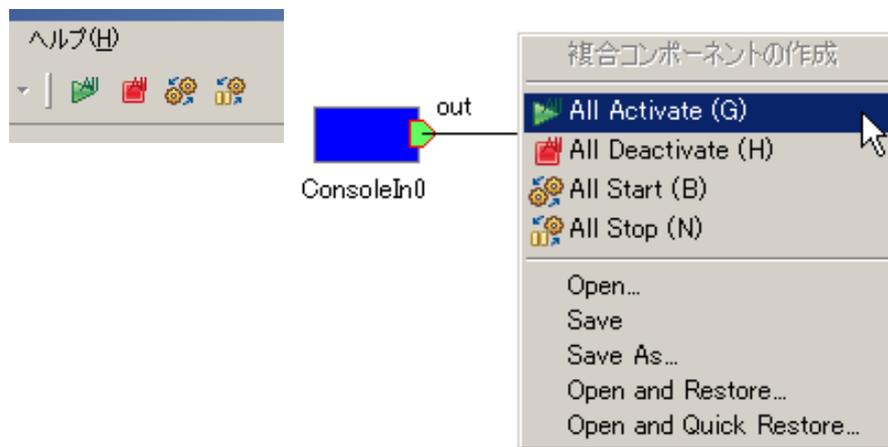
# RTコンポーネントの動作

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)

Connector Profile を入力してください。

Name: ConsoleIn0\_out\_ConsoleOut0.in

Data Type: IDLRTC/TypedLong:1.0

Interface Type: corba\_cdr

Dataflow Type: push

Subscription Type: flush

Push Rate(Hz):

Push Policy: all

Skip Count:

詳細

Buffer (Output)

Buffer length: 8

Buffer full policy: overwrite

Buffer write timeout: 1.0

Buffer empty policy: readback

Buffer read timeout: 1.0

Buffer (Input)

Buffer length: 8

Buffer full policy: overwrite

Buffer write timeout: 1.0

Buffer empty policy: readback

Buffer read timeout: 1.0

Name	Value

追加 削除

OK キャンセル

- ※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

## ■ カメラコンポーネントの起動

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

## ■ ビューワコンポーネントの起動

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

## ■ 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

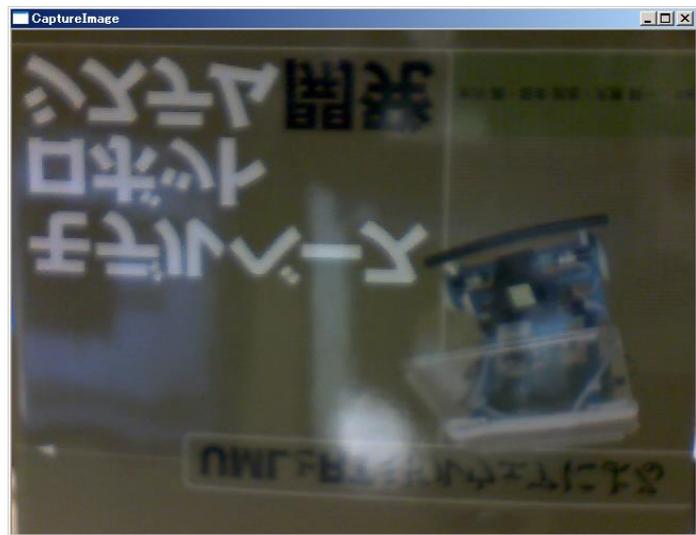
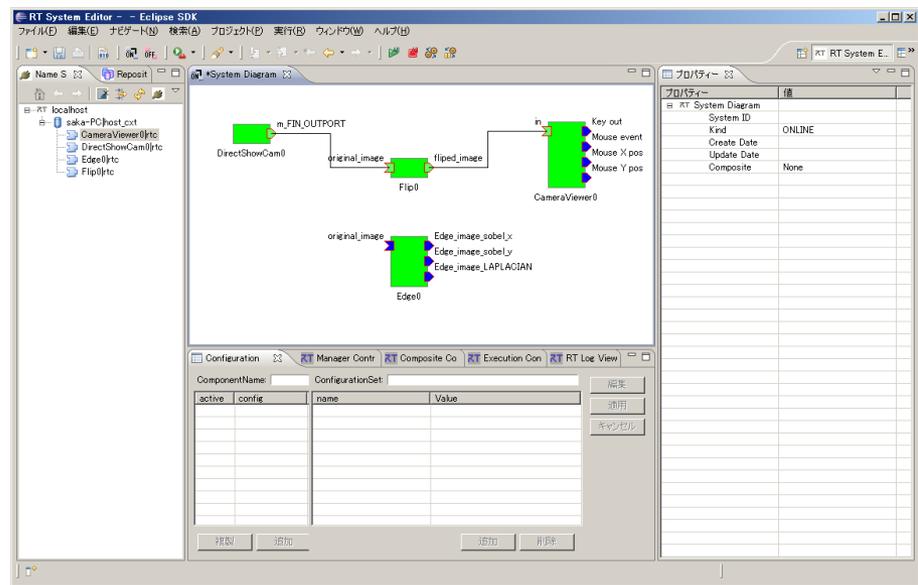
[プログラム] → [OpenRTM-aist 1.1] → [C++] → [components]  
→ [opencv-rtcs] → [FlipComp.exe]

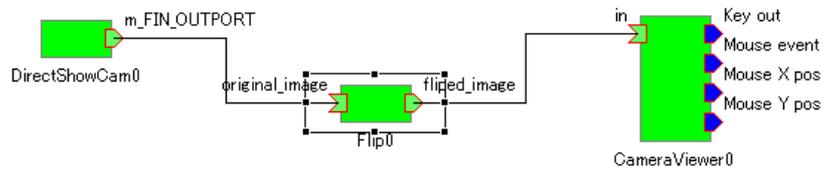
### ■ [スタート]メニューから起動

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

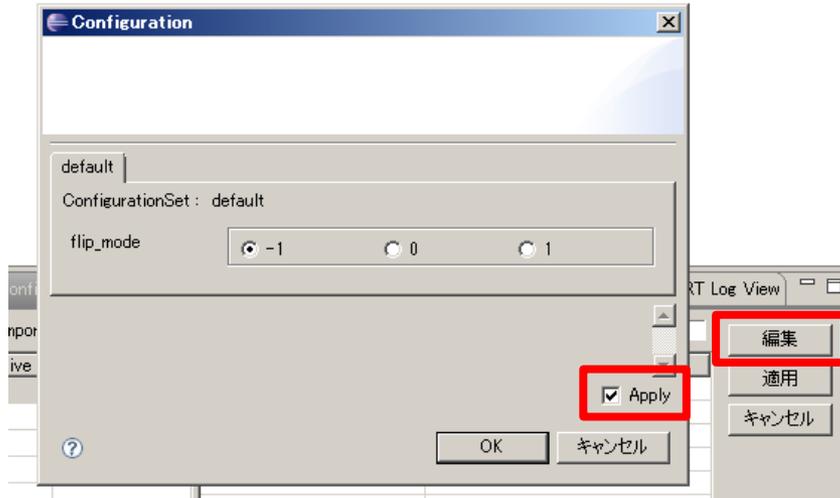
## Flip側との接続

- 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





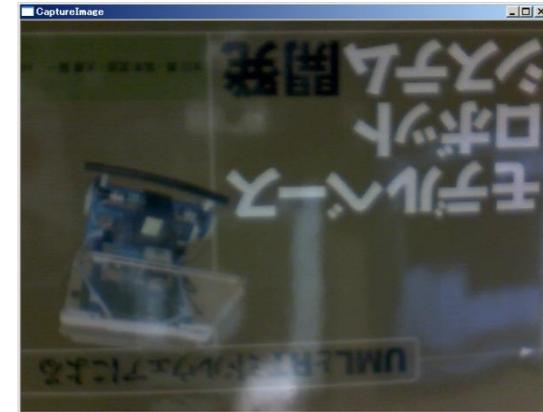
- ConfigurationViewの「編集」
- 表示されたダイアログ内で「flip\_mode」の値を変更
- 「Apply」のチェックボックス



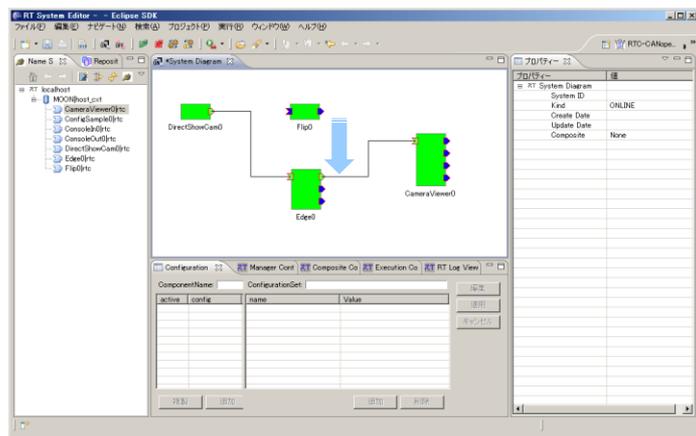
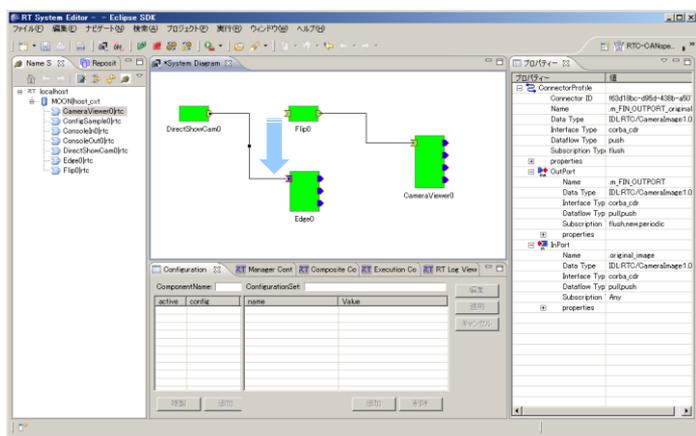
flip\_mode=1

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



# ネットワーク上の別のPCで動作するRTCの利用(1)

アクセス可能なネットワーク上に存在する別のPCで動作するネーミングサービスにアクセス

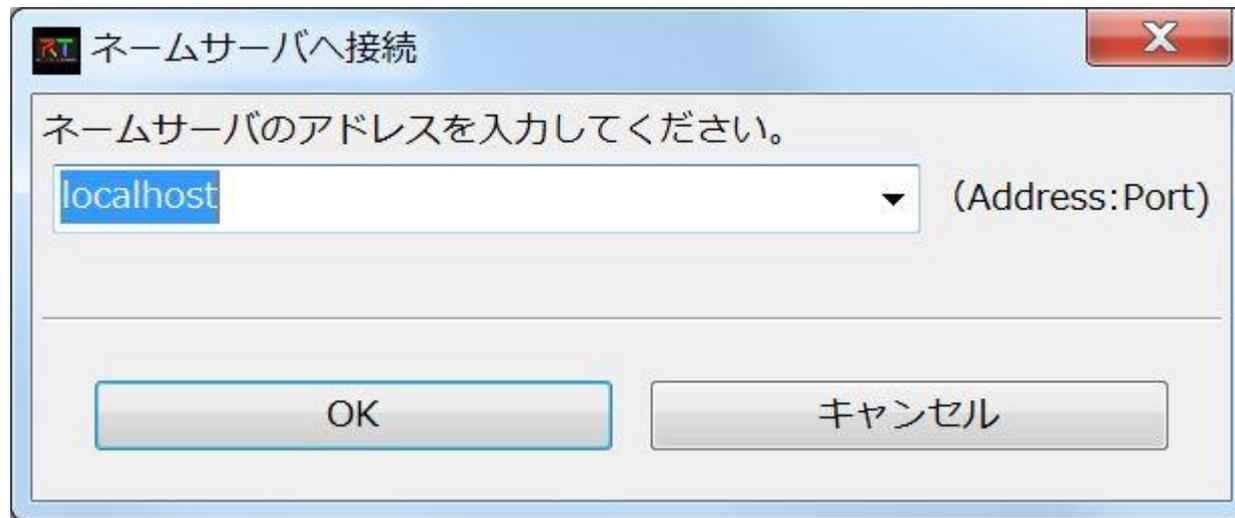
The screenshot shows the RT System Editor RCP interface. On the left, a tree view shows a project named 'localhost' with a sub-project 'rsdlab-vm-01|host...'. Under this sub-project, there are components 'CameraViewer0|rt...', 'DirectShowCam0|...', and 'Flip0|rtc'. A red box highlights a button in the toolbar above the tree view. A blue callout bubble points to this button, containing the text 'コンセントのボタンを押す。' (Press the consent button.).

On the right side of the interface, there is a 'プロパティー' (Properties) window for the selected component 'Flip0'. It lists various properties such as Path URI, Instance Name, Type Name, Description, Version, Vendor, Category, State, and properties, with their corresponding values.

At the bottom, there is a 'Configuration View' for the component 'Flip0'. It shows a table with columns 'name' and 'Value'. The table contains one row: 'flipMode' with a value of '0'. There are buttons for '編集' (Edit), '適用' (Apply), and 'キャンセル' (Cancel) next to the table.

# ネットワーク上の別のPCで動作するRTCの利用(2)

参照したいネーミングサービスが起動しているPCのIPアドレスとポートを入力する。



## IPアドレスの確認方法

コマンドプロンプトにおいて、「ipconfig」と入力する。

**他のPCで起動しているコンポーネントの閲覧およびRTCの遠隔利用ができる！**

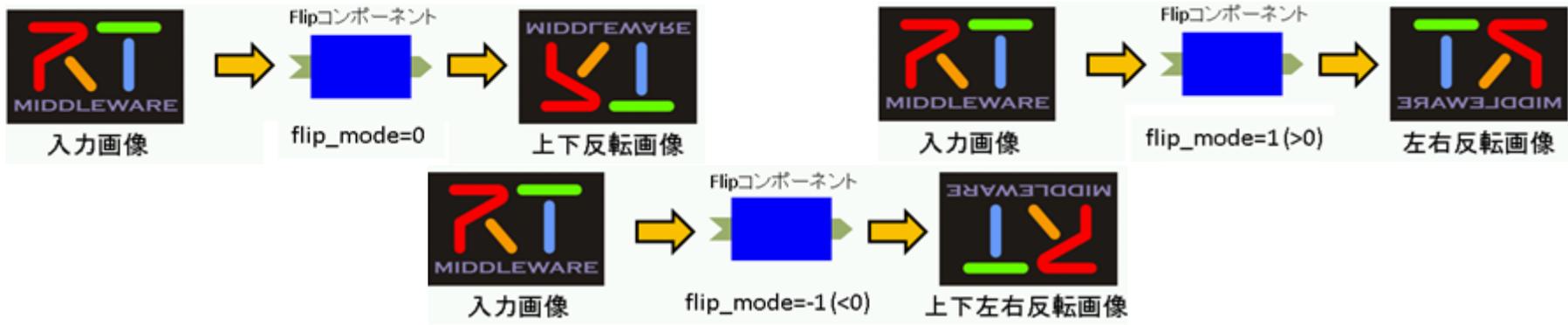
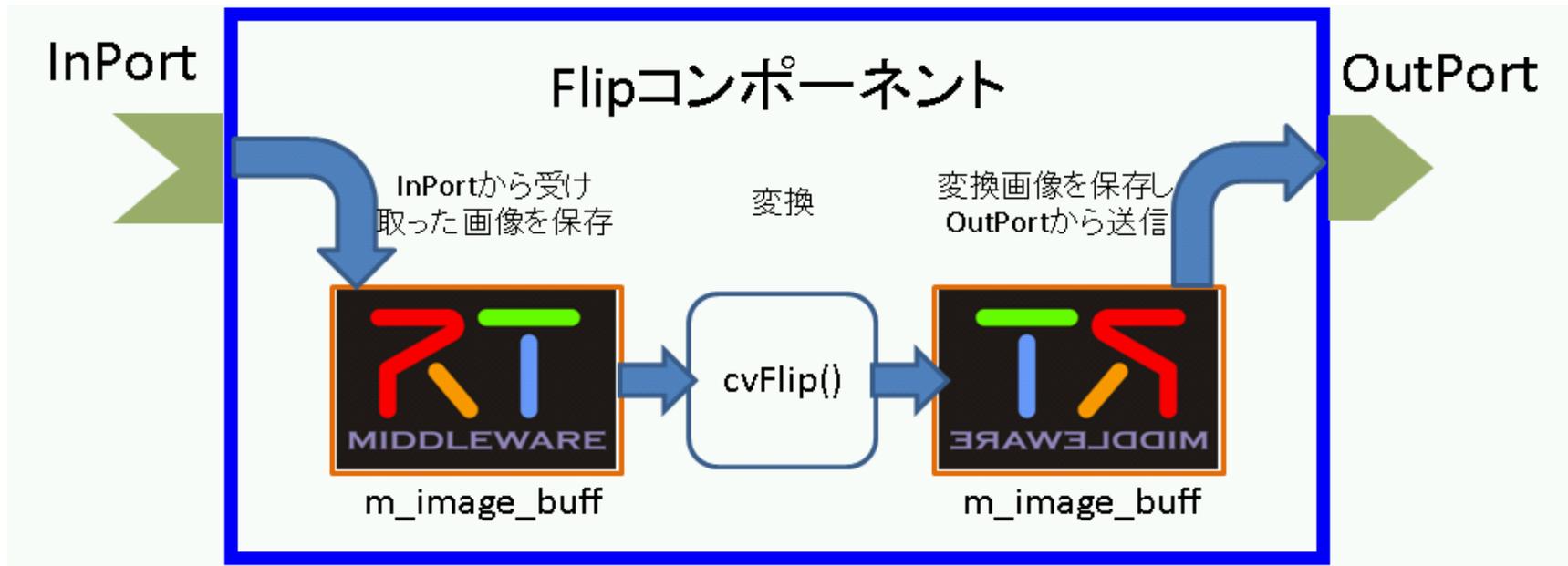
**(ファイヤーウォールがある場合は見えません(利用できない))**

# コンポーネント開発ツール RTCBuilderについて



# 第3部での演習の内容

- 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



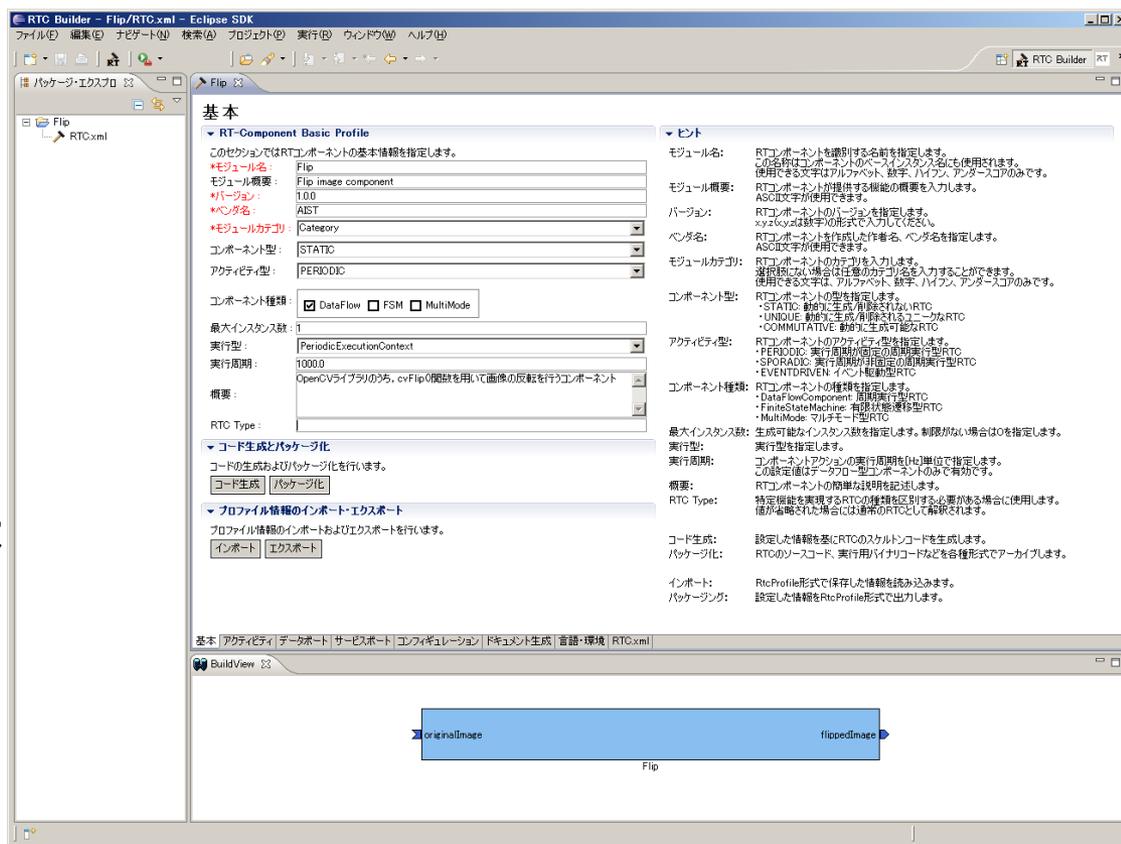
# RTCBuilder概要

## ■ 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 displays the RTC Builder application window, titled "RTC Builder - Flip/RTC.xml - Eclipse SDK". The interface is divided into several main sections:

- Package Explorer (パッケージ・エクスプローラ):** Located on the left, it shows a project named "Flip" containing an "RTC.xml" file.
- Basic Profile (基本):** The central configuration area, titled "RT-Component Basic Profile". It contains fields for:
  - Module Name (\*モジュール名): Flip
  - Module Summary (モジュール概要): Flip image component
  - Version (\*バージョン): 1.0.0
  - Vendor Name (\*ベンダ名): AIST
  - Module Category (\*モジュールカテゴリ): Category
  - Component Type (コンポーネント型): STATIC
  - Activity Type (アクティビティ型): PERIODIC
  - Component Kind (コンポーネント種類):  DataFlow  FSM  MultiMode
  - Maximum Instance Count (最大インスタンス数): 1
  - RTC Type (RTC Type):
- Code Generation and Packaging (コード生成とパッケージ化):** Includes buttons for "Code Generation" (コード生成) and "Packaging" (パッケージ化).
- Profile Information Import/Export (プロファイル情報のインポート・エクスポート):** Includes buttons for "Import" (インポート) and "Export" (エクスポート).
- Hints (ヒント):** A detailed list of configuration options on the right side, such as "Module Name", "Module Summary", "Version", "Vendor Name", "Module Category", "Component Type", "Activity Type", "Component Kind", "Maximum Instance Count", "Execution Type", "Execution Period", "Summary", "RTC Type", "Code Generation", "Packaging", "Import", and "Packaging".
- Build View (ビルドビュー):** Located at the bottom, it shows a diagram of the component "Flip" with two data flows: "originalImage" and "flippedImage".

パッケージ・エクスプローラ

RTCプロファイルエディタ

ヒント

ビルドビュー

# 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

# 準備

- 初期画面のクローズ
  - 初回起動時のみ

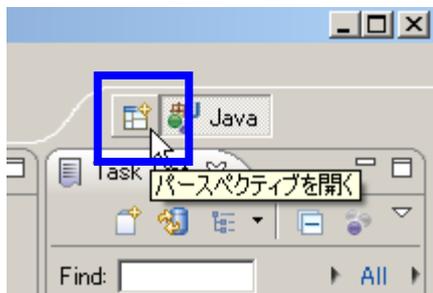


## ※パースペクティブ

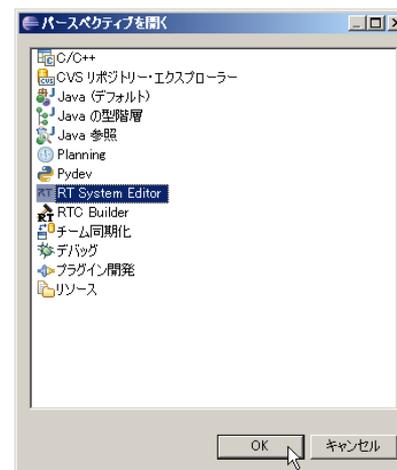
Eclipse上でツールの構成を管理する単位  
メニュー, ツールバー, エディタ, ビューなど  
使用目的に応じて組み合わせる  
独自の構成を登録することも可能

## ■ 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

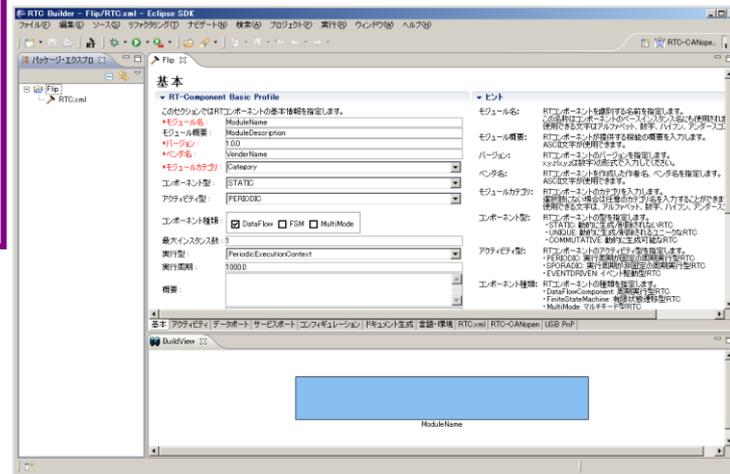
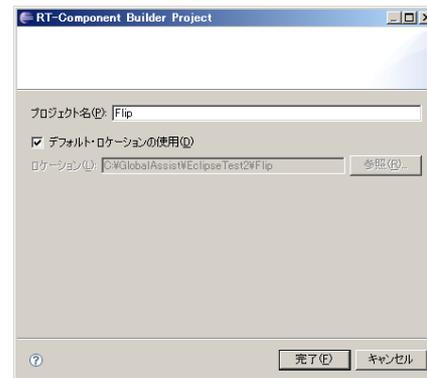
②にて「デフォルト・ロケーションの使用」チェックボックスを外す

「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」



# 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

# 基本プロフィール

## ■ 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 'RT-Component Basic Profile' configuration window. The '基本' (Basic) section is expanded, showing various fields for configuring the component. The 'ヒント' (Hint) section provides additional information about the fields. A blue box highlights the following configuration details:

- 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

# アクティビティ

## ■ 生成対象RTCで実装予定のアクティビティを設定

**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。

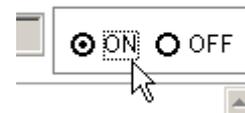
**事前条件:** アクティビティを実行する前に成立すべき事前条件を記述します。

**事後条件:** アクティビティを実行した後に成立すべき事後条件を記述します。

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

▼ヒント

データポート: RTコンポーネント間でデータを送受信するOutPortとInPortを接続する。

InPort: RTコンポーネントに他のRTコンポーネントからデータを受け取る。

OutPort: RTコンポーネントから他のRTコンポーネントにデータを送信する。

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内で一意なASCII文字が使用できる。

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依存する。

ポートの場所: RTSystemEditorなどのこのプロバイダはオプションでポートの場所を指定することができます。

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、このレベルの情報を記述してください。

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。

② 設定する型情報を一覧から選択

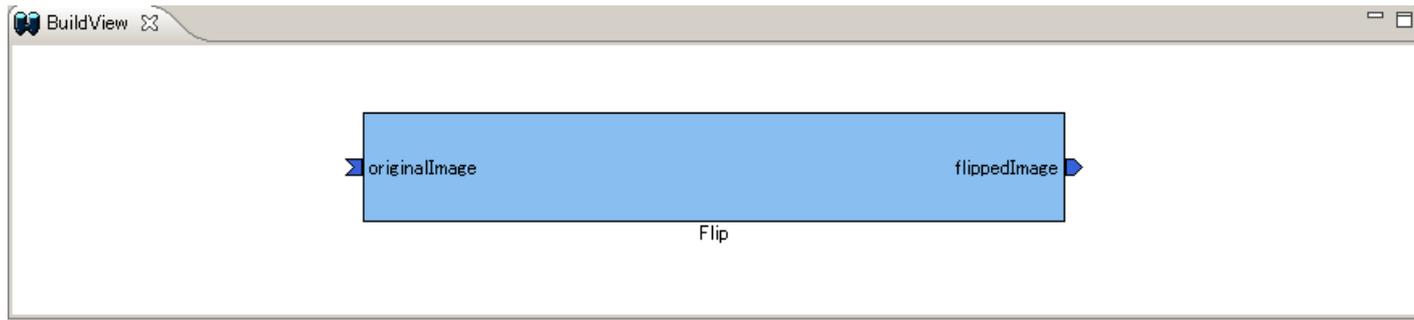
※ データ型は、型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

※ 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

※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

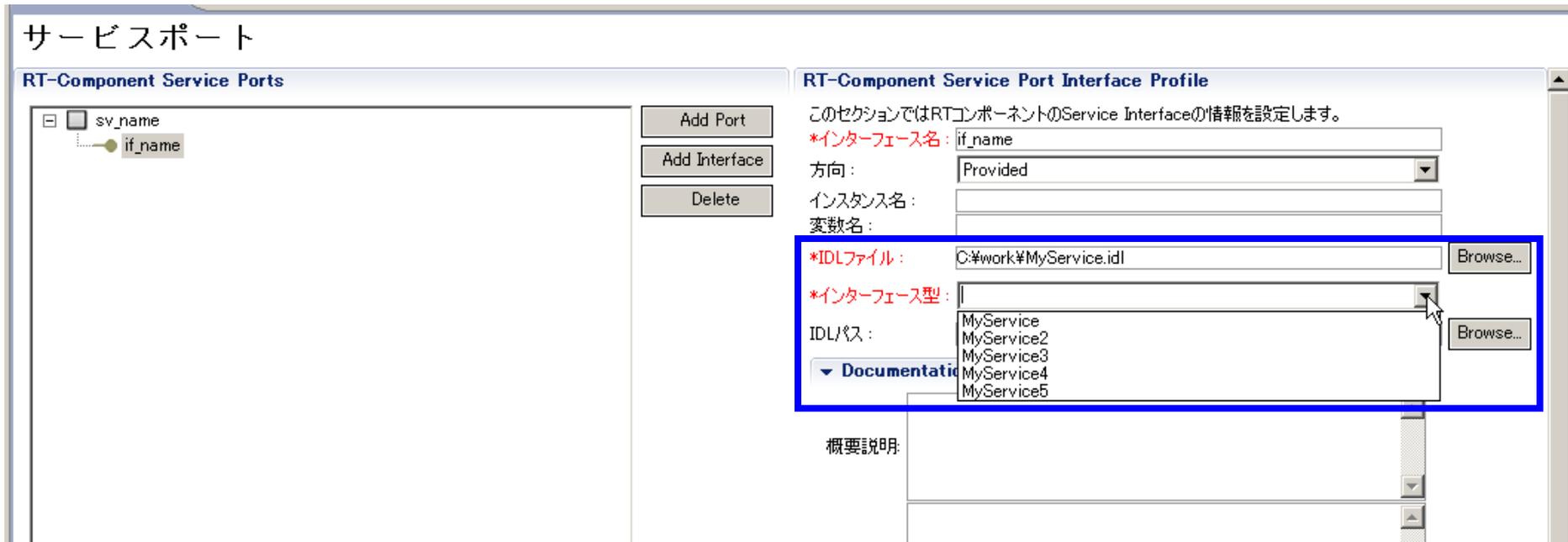
ポート名: **flippedImage**

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

▼ 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	Add	Delete
flipMode		

▼ 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<BR>1: 左右反転したい場合<BR>-1: 上下左右反転したい場合

▼ ヒント  
Config. Param: RTコン  
コンフ  
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コン  
解釈  
変数名: コンフ  
実除  
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

▼ 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	Add	Delete
conf_name0		

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.

### ▼ 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

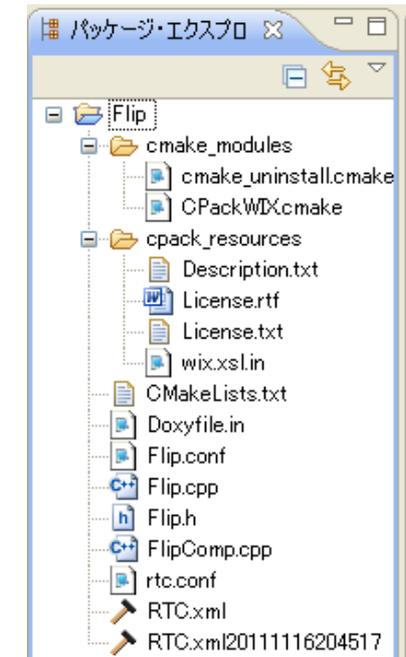
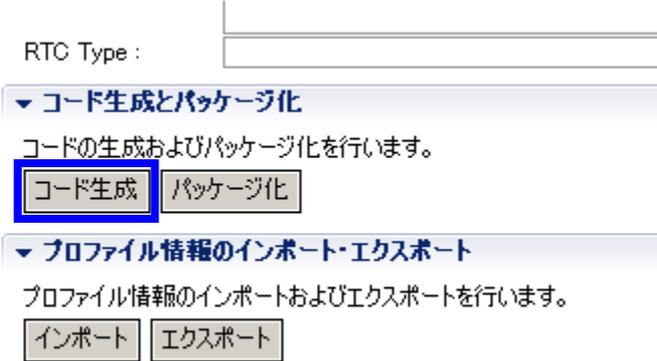
### ▼ ヒント

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、プロフィール内にもみ保存されます。

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

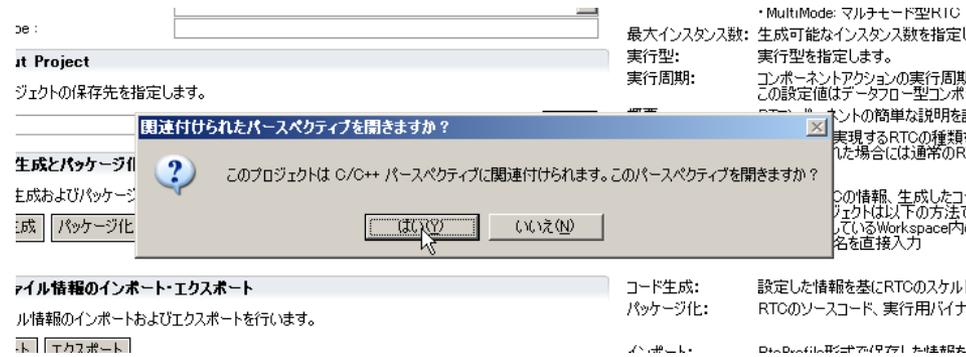
**「C++」を選択**

## ■ 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

(EclipseにCDTなどの開発ツールがインストールされている場合)

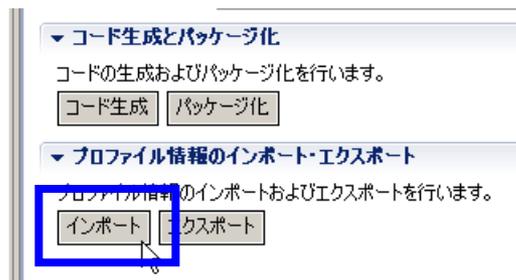


※生成コードが表示されない場合には、「リフレッシュ」を実行

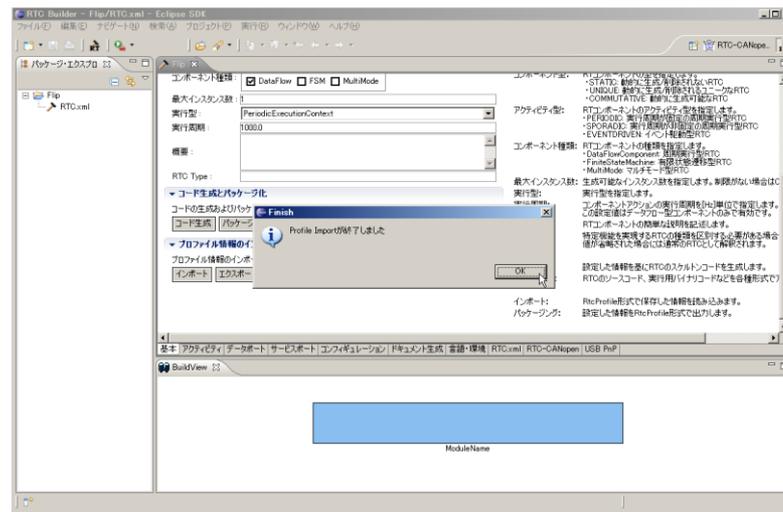
C++版RTC → CDT  
 Java版RTC → JDT  
 (デフォルトインストール済み)  
 Python版 → PyDev

# 既存のRTCの設定(RTC Profile)を利用したい場合

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

# おわりに

- 第2部では、RTコンポーネント開発とRTコンポーネントを用いたシステム構築に必要なツールであるRT System Editorの使い方を体験した.
- RTC BuilderやRT System Editorについては、産総研原氏によりブラウザ上で動作するバージョンが開発が進められている.

<http://openrtp.org/r tcbow/index.html>

[http://hara.jpn.com/siwiki/\\_hara/ja/Software/RTSEoW.html](http://hara.jpn.com/siwiki/_hara/ja/Software/RTSEoW.html)

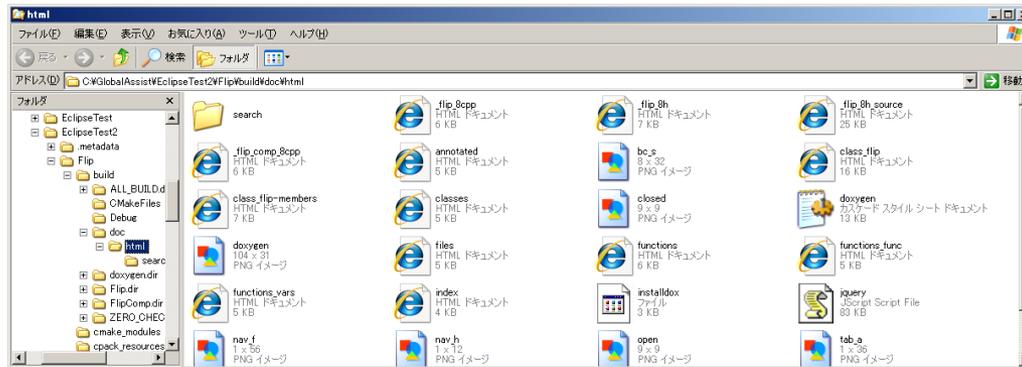
- RT System Editorを用いたシステム構築は初期段階での運用には適しているが、実運用段階では、rtshellなどのRTシステムの自動構築を可能にするツールの利用が好ましい.

<http://openrtm.org/openrtm/ja/node/869>

# RTCBuilder補足説明



## ※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
生成	生成索引	生成メンバ
<b>クラス Flip</b>		
Flip image component. [詳細]		
<a href="#">#include &lt;Flip.h&gt;</a>		
すべてのメンバ一覧		
<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	

### 説明

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のコアフィギュレーション機能を使用してFlipModeという名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。  
 ・上下反転したい場合、0  
 ・左右反転したい場合、1  
 ・上下左右反転したい場合、-1

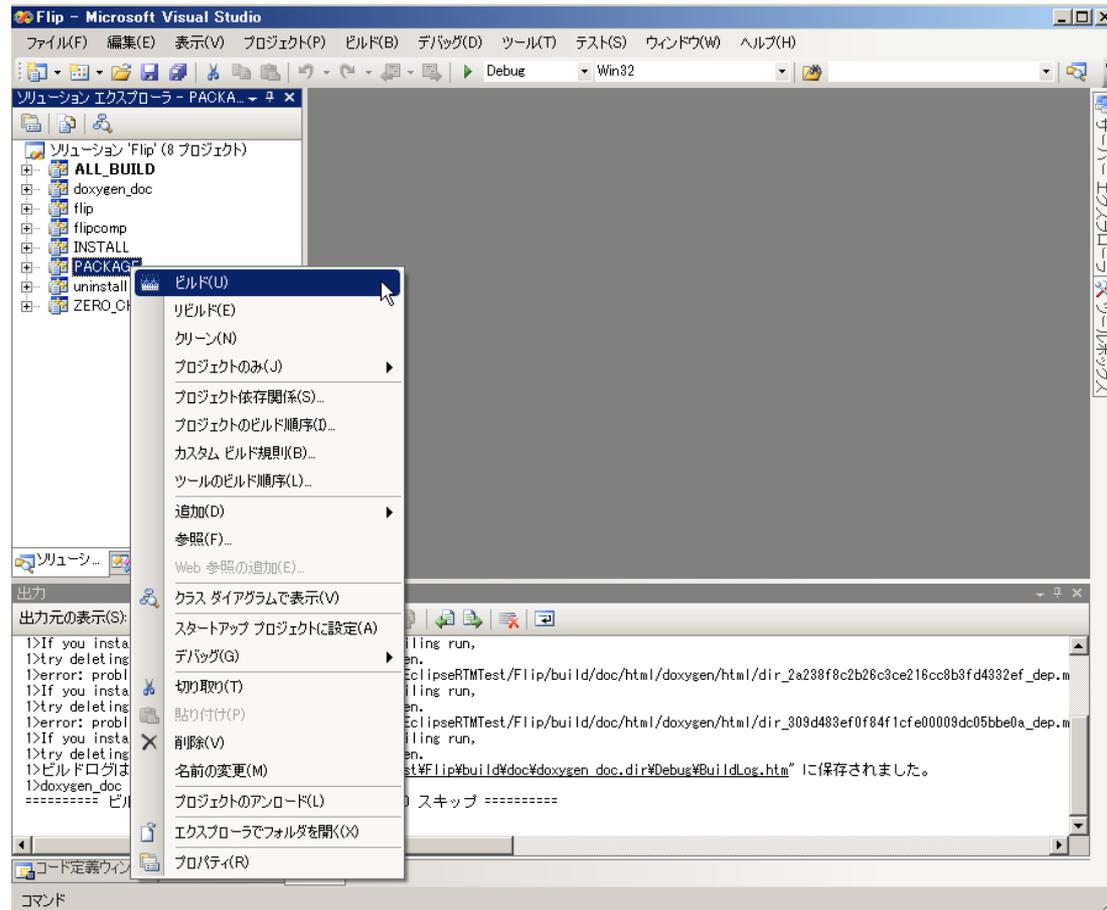
作成するRTCの入出力仕様は以下のとおりです。

flip 1.0.0

メインページ	クラス	ファイル
<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;rtm/Manager.h&gt; 00028 #include &lt;rtm/DataFlowComponentBase.h&gt; 00029 #include &lt;rtm/CorbaPort.h&gt; 00030 #include &lt;rtm/DataInPort.h&gt; 00031 #include &lt;rtm/DataOutPort.h&gt; 00032 #include &lt;rtm/idl/BasicDataTypeSkel.h&gt; 00033 #include &lt;rtm/idl/ExtendedDataTypesSkel.h&gt; 00034 #include &lt;rtm/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 00091 ~Flip(); 00092 00093 // &lt;rtc-template block="public_attribute"&gt; 00094 00095 // &lt;/rtc-template&gt; 00096 00098 // &lt;/rtc-template block="public_attribute"&gt;                     </pre>		

関数
RTC::ReturnCode_t Flip::onActivated ( RTC::UniqueId ec_id ) [virtual]
データ精度の確保 ・イメージ用メモリの初期化 ・OutPortの画面サイズの初期化
RTC::ReturnCode_t Flip::onDeactivated ( RTC::UniqueId ec_id ) [virtual]
データ精度の解放 ・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の画像データを内部バッファにコピー ・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
<ul style="list-style-type: none"> <li>Name: flipMode flipMode</li> <li>DefaultValue: 0</li> <li>Unit: なし</li> </ul>

## ■ ソリューション中の「PACKAGE」をビルド

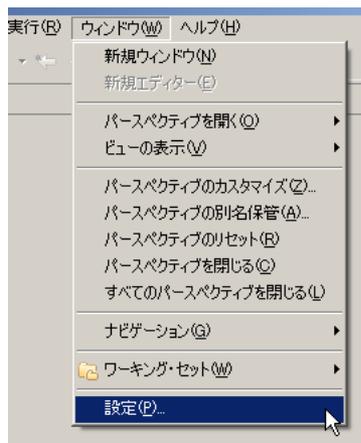


- binaryにて指定したディレクトリ直下にmsi形式のインストールパッケージを生成
  - コンポーネントのインストール先  
C:¥Program Files¥OpenRTM-aist¥1.1¥components¥<言語>/<パッケージ名>

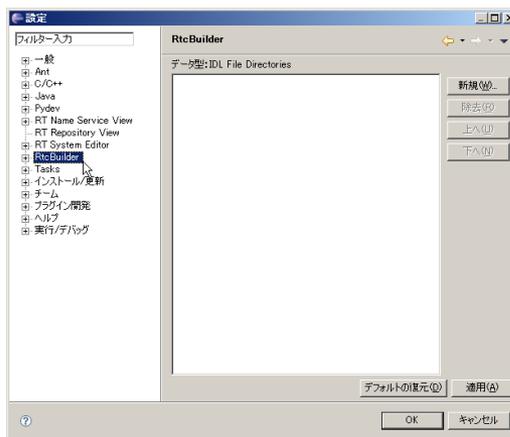
## ■ DataPortにて利用するデータ型の指定

→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

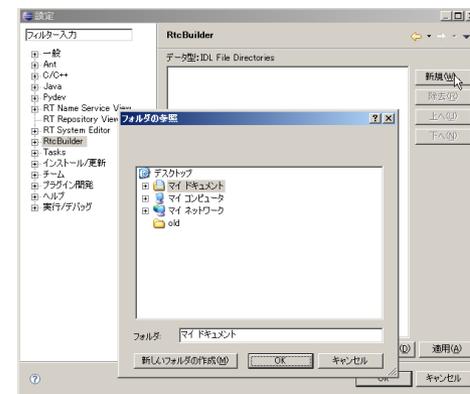
①メニューから「ウインドウ」→「設定」



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

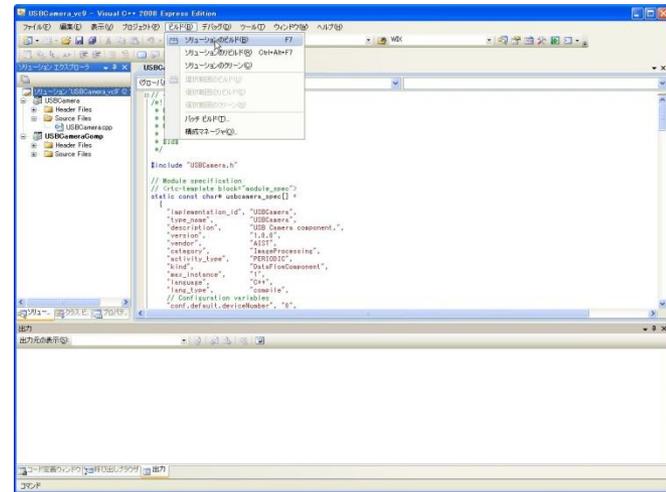
→BasicDataType.idl, ExtendedDataTypes.idlなど

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

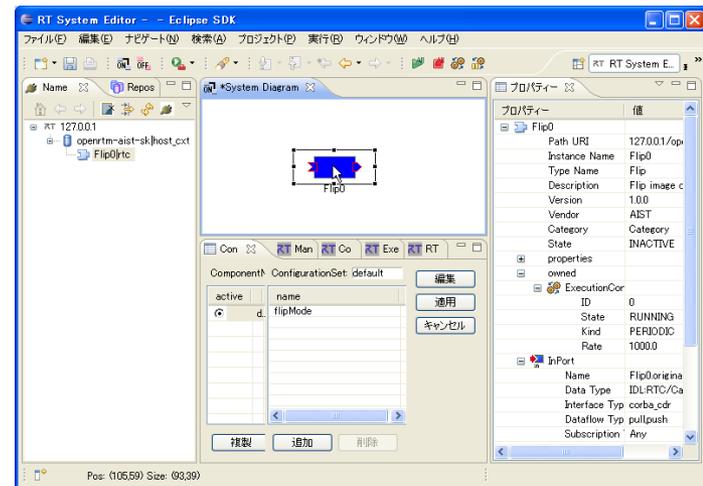
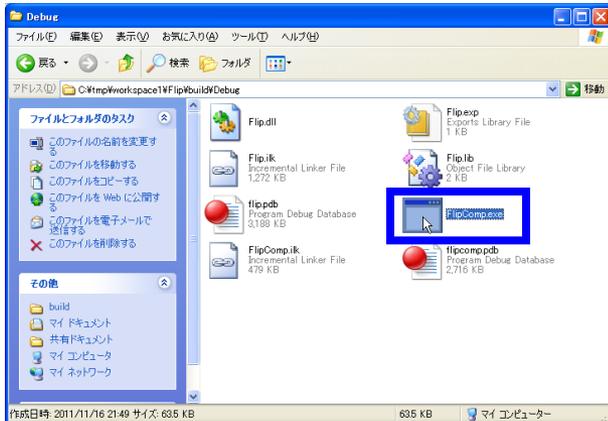
①コード生成先ディレクトリ内の「copyprops.bat」をダブルクリックして、設定ファイルをコピー



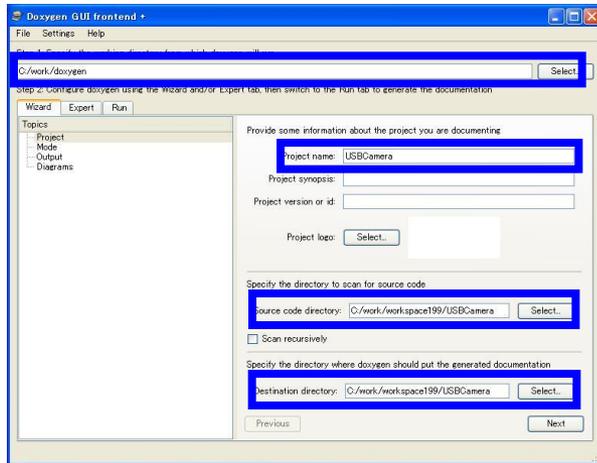
②VisualStudioを用いたビルド



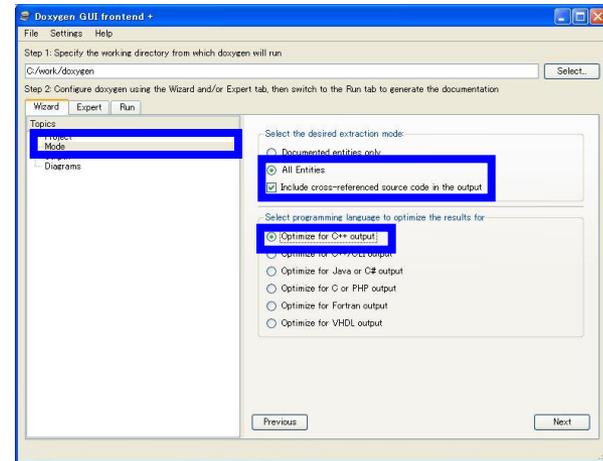
③FlipComp¥¥Debug内のFlipComp.exeを起動



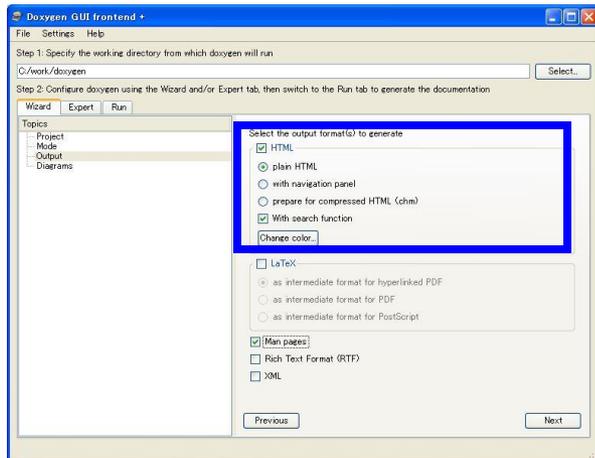
## ① Doxygen用GUIツールを起動 作業用ディレクトリ,ソース格納場所, 生成ファイル出力先,プロジェクト名を指定



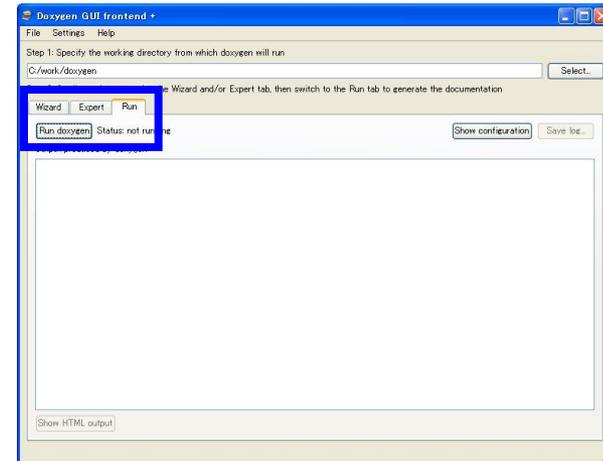
## ② 「Mode」セクションにて, 出力内容,使用言語を指定



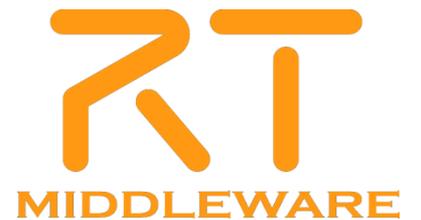
## ③ 「Output」セクションにて, html出力を指定



## ③ 「Run」タブにて, 「Run doxygen」を実行



# RTSystemEditor補足説明

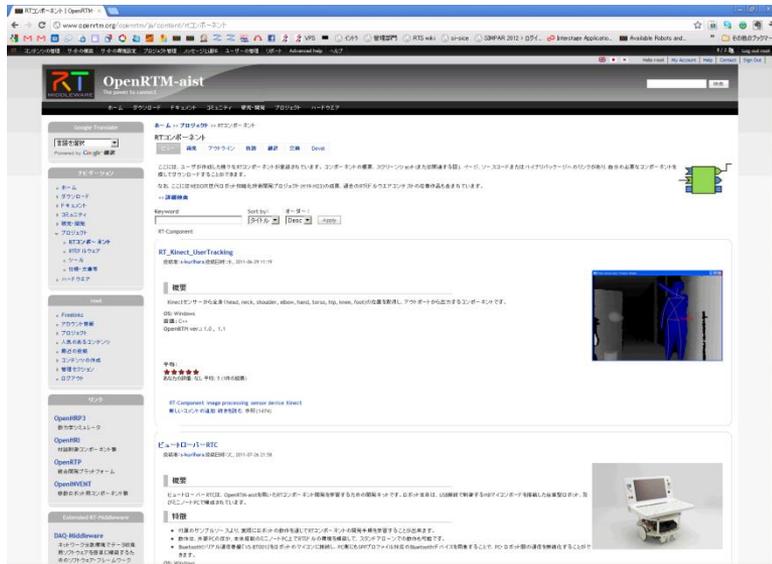


## ■ プロジェクトとは

- 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。



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

## ■ 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

The screenshot shows the OpenRTM-aist website interface. At the top, there are language selection icons (UK, Germany, Japan, Korea) and user links (Hello, My Account, Help, Contact, Sign Out). The main header features the OpenRTM-aist logo and tagline 'The power to connect'. Below the header is a navigation menu with links for Home, Downloads, Documents, Community, Research/Development, and Projects.

The main content area displays the '顔検出コンポーネント' (Face Detection Component) page. It includes a breadcrumb trail: Home >> Project >> RT Component >> Face Detection Component. The page title is '顔検出コンポーネント' and the author is 's-kurihara' with a posting date of '2011-11-15 18:20'. The contact email is 'openrtm@openrtm.org'. A description in Japanese explains that the component takes an image from the InPort and outputs a face-detection result image from the OutPort, including face position and count. A screenshot of the 'CapturedImage' window shows a person's face with a red bounding box.

Below the description is a 'Downloads' section with a table:

バージョン	Downloads	日付	Links
0.1	<b>Download (17.44 MB)</b>	2011-11-15	Notes

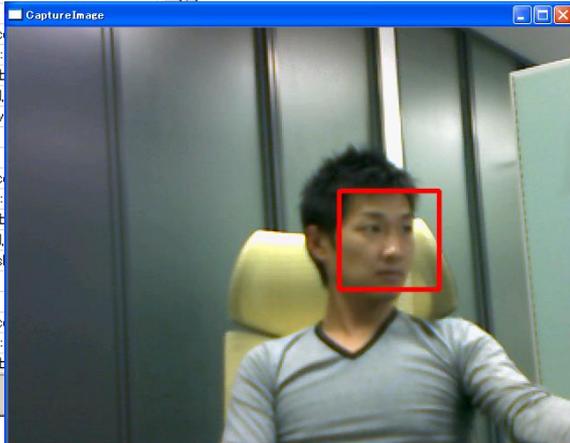
The 'Download (17.44 MB)' link is highlighted with a red box. There are also links for 'View all releases' and 'View pending patches'.

On the right side of the page, there is an 'Issues' section with a search bar and a '高度な検索' (Advanced Search) button. Below it, it shows 'All issues: 0 open, 0 total' and 'Bug reports: 0 open, 0 total'. There is also a 'ユーザーログイン' (User Login) section with fields for 'ユーザー名' (Username) and 'パスワード' (Password), and a 'ログイン' (Login) button. Below the login fields are links for 'アカウントの作成' (Create Account) and 'パスワードの再発行' (Reset Password).

- 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

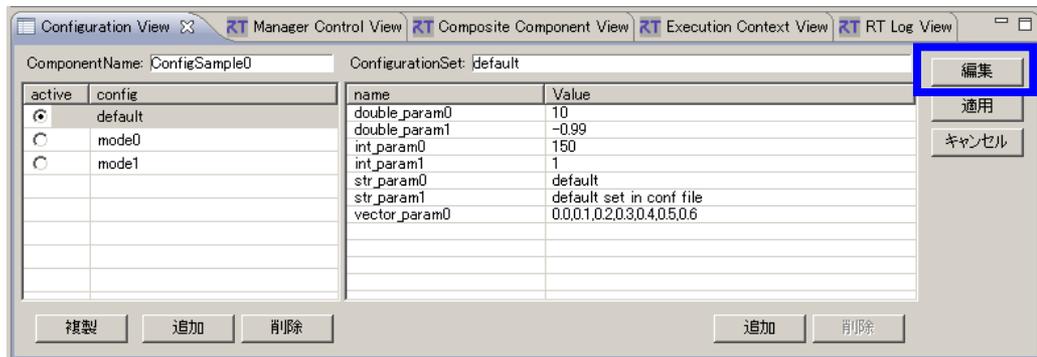
The screenshot shows the RT System Editor interface. The main window displays a system diagram with components: DirectShowCam0, Flip0, FaceDetect0, Edge0, and CameraViewer0. The configuration panel for FaceDetect0 is visible, showing parameters like downscale, haarcascade, min\_object\_height, and min\_object\_width.

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



- 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

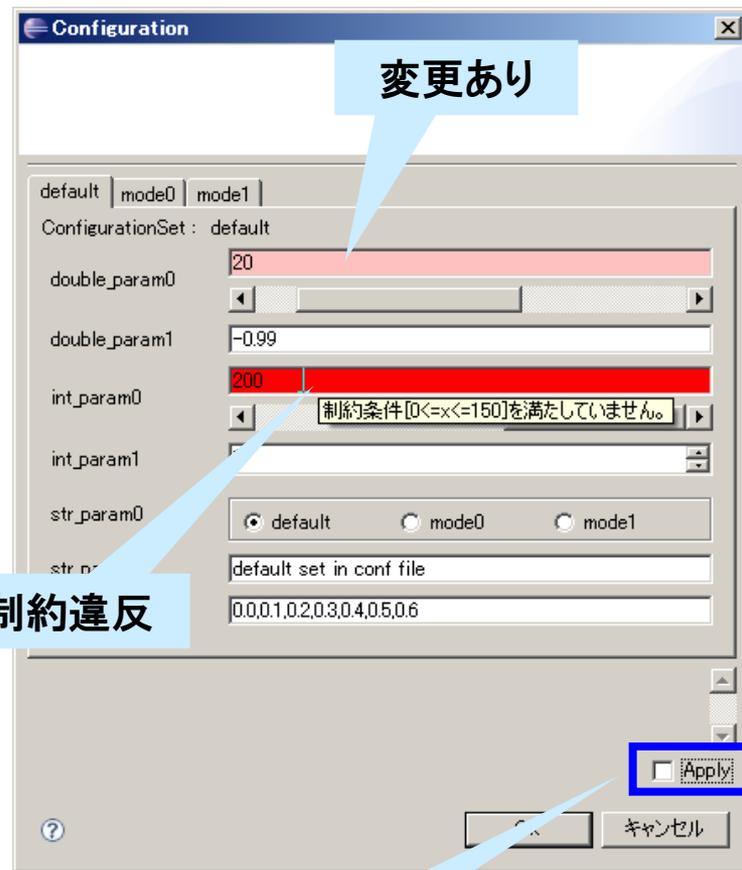
## RTコンポーネントのコンフィギュレーション情報の確認/編集



※「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



変更あり

制約違反

即時反映

- 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)

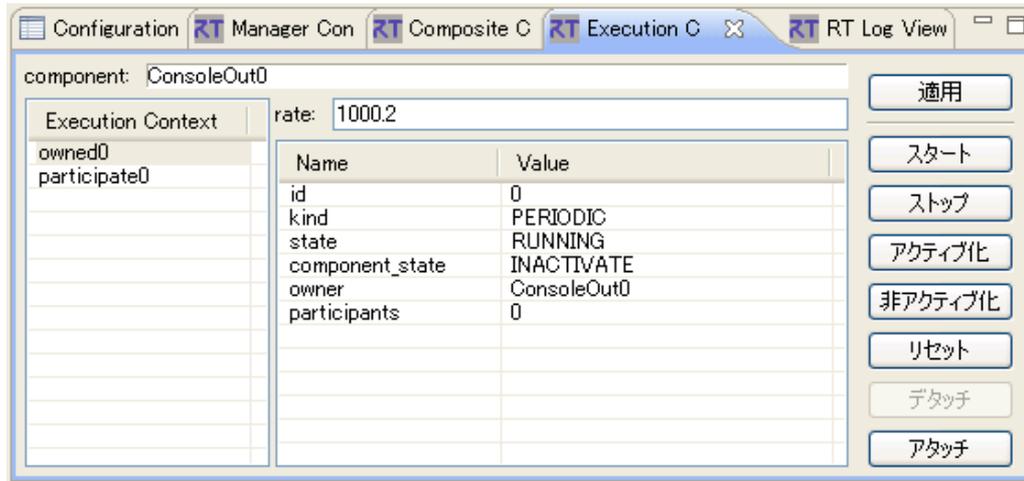
conf. \_\_[コンフィグセット名]. [コンフィグパラメータ名] : [制約情報]

※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

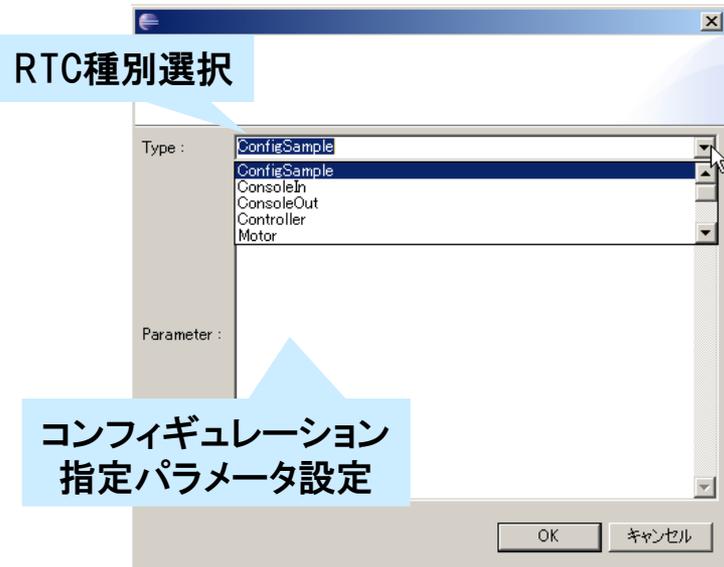
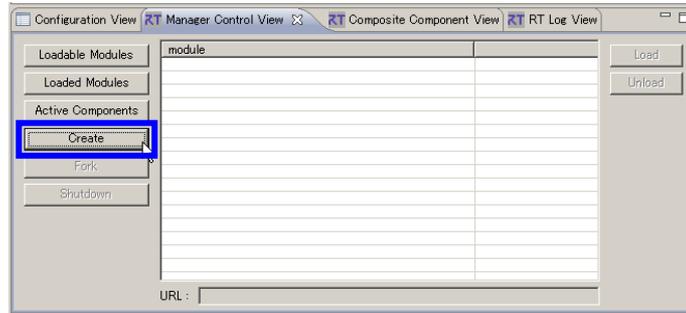
■ RTコンポーネントが属する実行コンテキスト(EC)を一覧表示



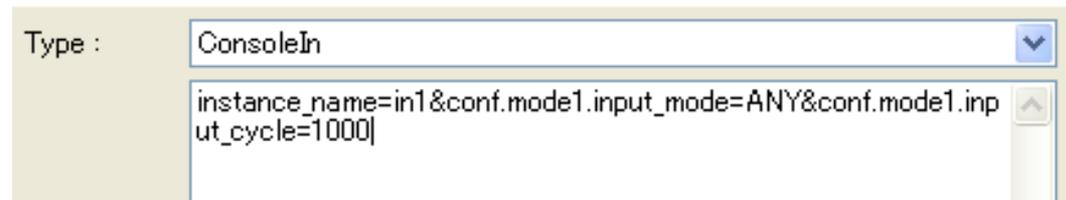
属性名	説明
id	ECのID. オンラインの場合には, context_handleを表示
kind	ECの種別(PERIODIC/EVENT_DRIVEN/OTHER)
state	ECの状態(RUNNING/STOPPING)
component state	対象RTCの状態(ACTIVE/INACTIVE/ERROR)
owner	対象ECを所有しているオーナーRTCのインスタンス名
participants	対象ECに参加中のRTCの数

※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



- **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

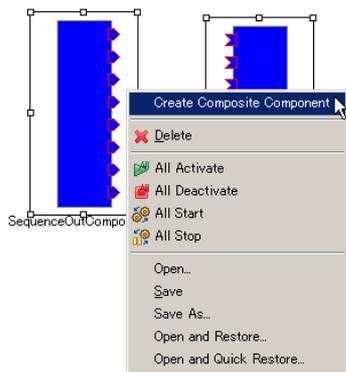




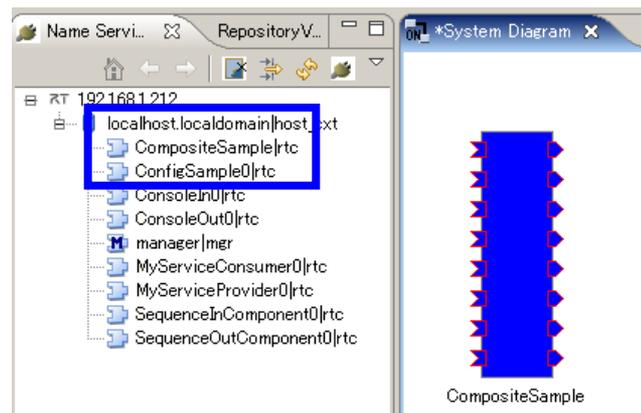
## ■ 複数のRTCをまとめて、1つのRTCとして扱うための仕組み

## ● 複合コンポーネントの作成方法

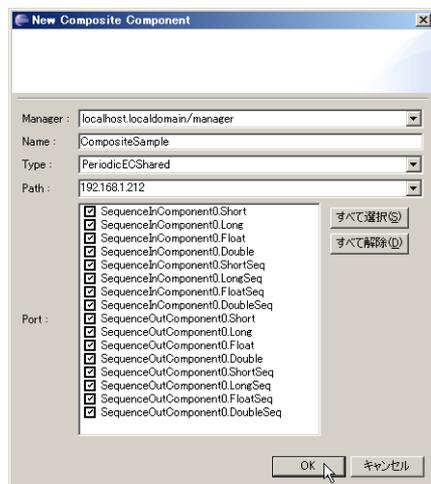
① 複数RTCを選択している状態で右クリック



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

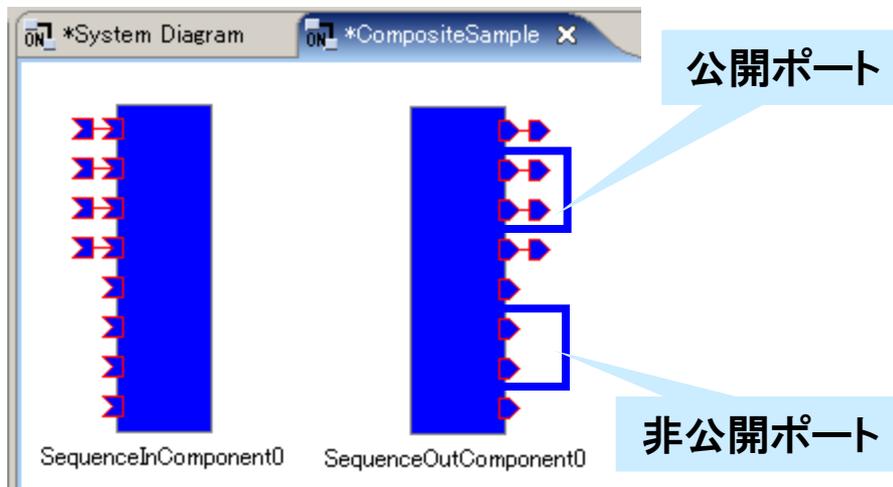
※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

## ■ 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

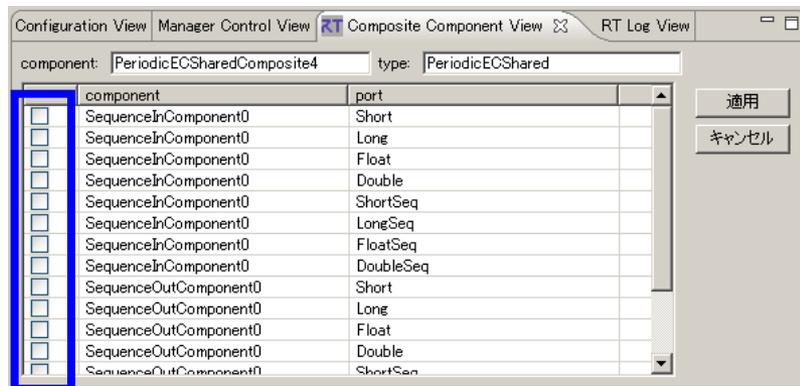


- ※エディタ内に別RTCをDnDすることで、子コンポーネントの追加が可能  
→追加したRTCのポートは全て非公開に設定
- ※エディタ内のRTCを削除することで、子コンポーネントの削除が可能  
→削除されたRTCは、親エディタに表示

## ■ 公開ポートの設定

### ● 複合コンポーネントビュー

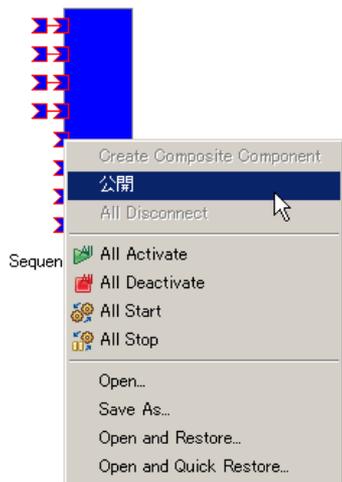
ポート公開情報



※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」



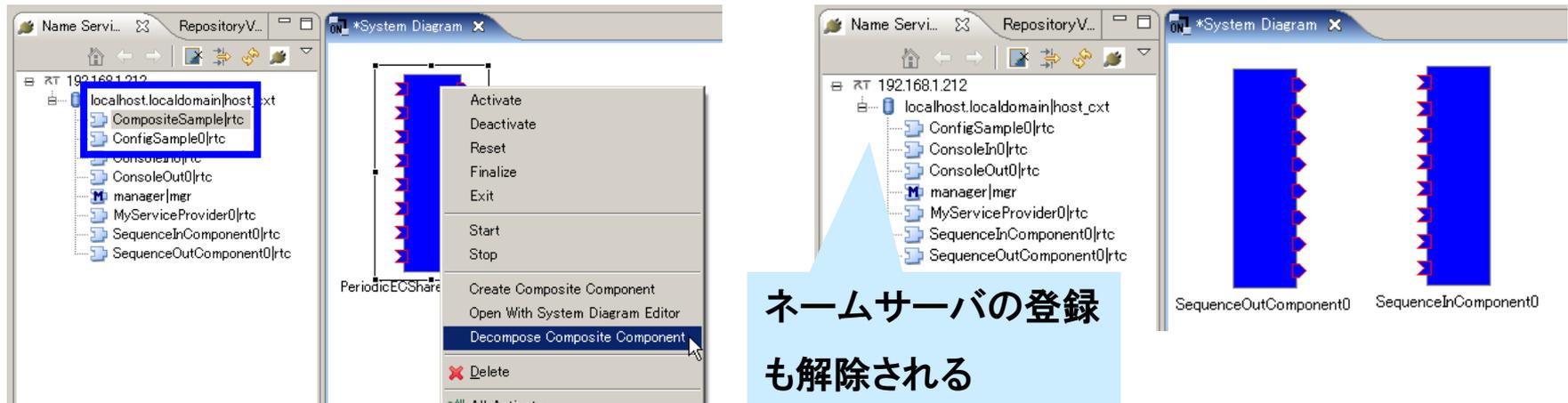
※公開ポートを「非公開」



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

- 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

The screenshot shows the RT System Editor interface. The main window displays an 'Offline System Diagram' with components: CameraComponent\_1, ImageProcess\_1, and ImageViewer\_1. Three callout boxes highlight key features: 'リポジトリビュー' (Repository View) on the left, 'オフライン・システムエディタ' (Offline System Editor) in the center, and 'プロパティビュー' (Property View) on the right. At the bottom, the 'Configuration View' is visible, showing a table for component configuration.

active	config	name	Value

On the right, the 'プロパティビュー' (Property View) shows the following details for ImageProcess\_1:

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 - 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き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

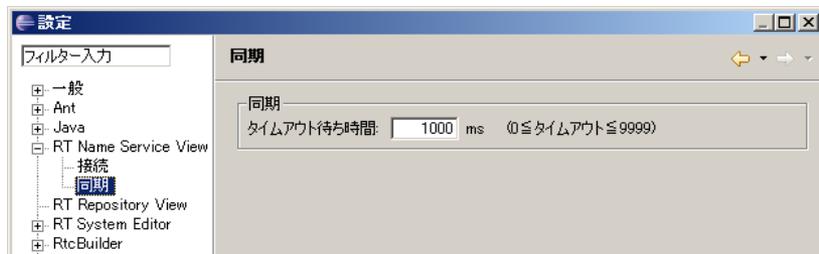
## ■ 「RT Name Service View」－「接続」【接続周期】

- 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



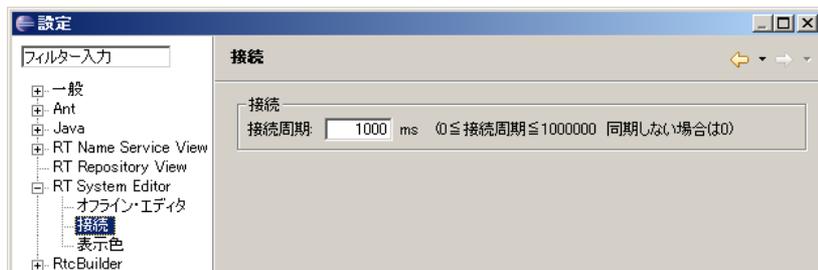
## ■ 「RT Name Service View」－「同期」【タイムアウト待ち時間】

- 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**

## ■ 「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

