

第2部

RTコンポーネント作成入門



第2部の目的



- 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)

OpenRTM-aistの開発支援ツール



- 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
- RTC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

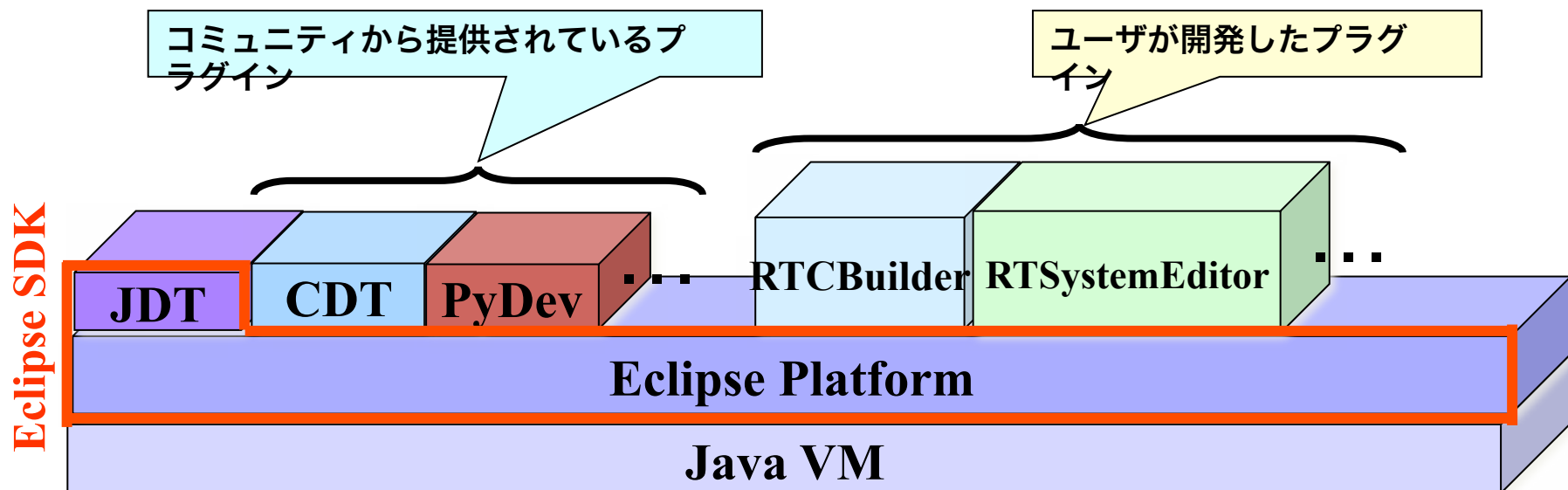
など



RTC BuilderとRT System Editor



- 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



RTC BuilderとRT System Editorの利用方法



- OpenRTM-aist-1.1.1のインストール時に、インストールされるので、そちらを利用。
「スタート」 → 「すべてのプログラム」 → 「OpenRTM-aist 1.1」 → 「Tools」にある、OpenRTPを選択。
- 単体でダウンロードする場合、Linuxの場合は、下記のサイトからダウンロードし、解凍。
(別途、JAVAの環境が必要になるので、注意。)

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" (All-in-one 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 "全部入りパッケージ" (All-in-one package) for Eclipse-3.4.2 [Ganymede SR2].

Eclipse-3.4.2 [Ganymede SR2]		
Eclipse3.4.2+RTSE+RTCB Windows用全部入り	eclipse342_rtmttools110-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, and a terminal snippet showing the installation process:

```
$ 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 amd64 XPCOM application runner
```



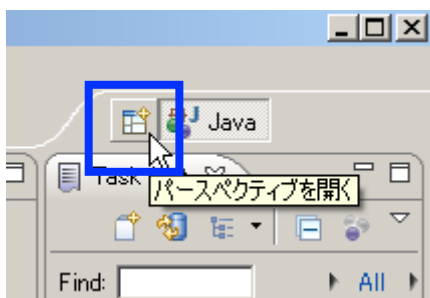


システム構築支援ツール RTSystemEditor

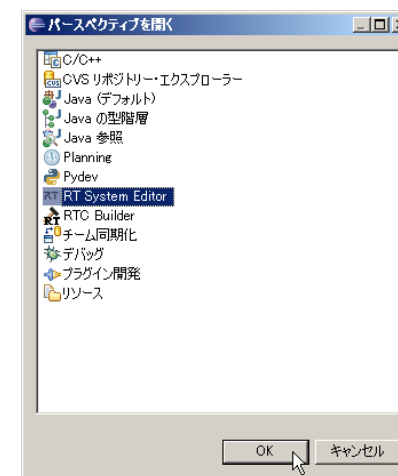


■ パースペクティブの切り替え

- 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

RT System Editorの概要



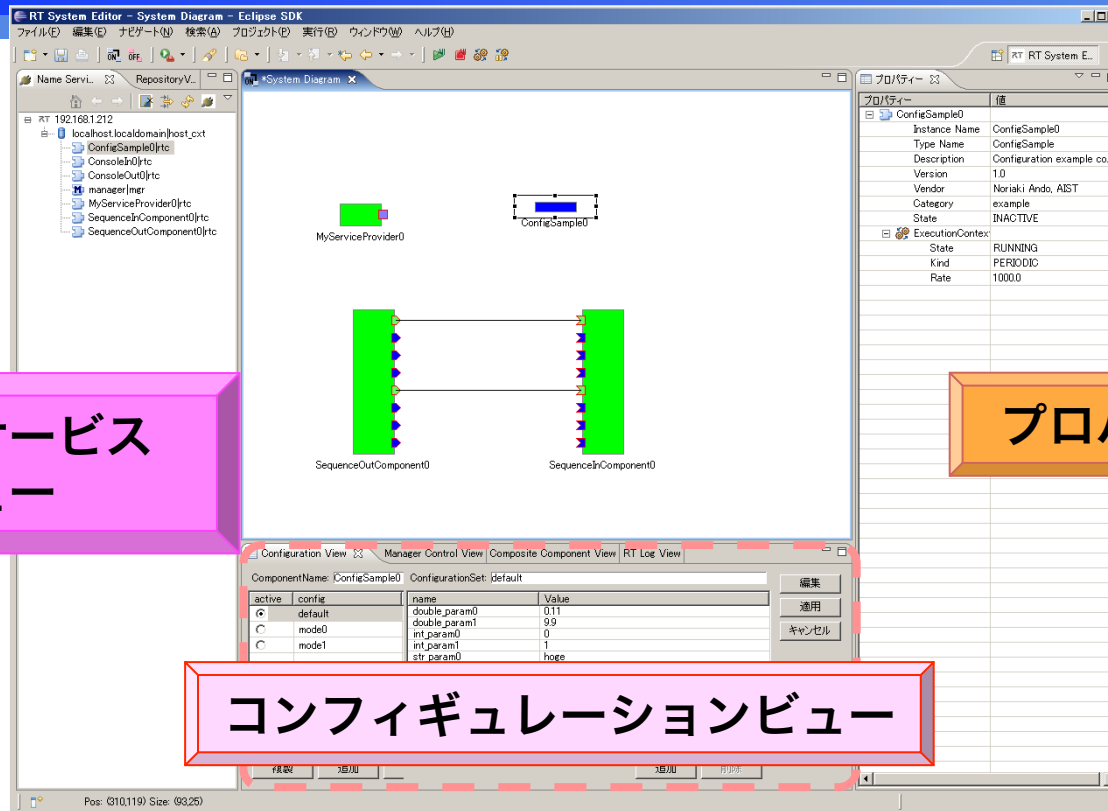
RTコンポーネントを組み合わせて、RTシステムを構築するためのツール。（Simlinkのようなイメージ。）

The screenshot displays the RT System Editor interface. On the left is a project tree showing components like `ConfigSample0`, `ConsoleIn0`, `ConsoleOut0`, `managerImr`, `MyServiceProvider0`, `SequenceInComponent0`, and `SequenceOutComponent0`. The main workspace shows a system diagram with `MyServiceProvider0`, `ConfigSample0`, `SequenceOutComponent0`, and `SequenceInComponent0` connected. On the right, the 'Properties' panel shows details for `ConfigSample0`, including its instance name, type, description, version, vendor (Noriaki Ando, AIST), category (example), and state (INACTIVE). Below the diagram, the 'Configuration View' for `ConfigSample0` is active, showing a table of configuration parameters:

active	config	name	Value
<input checked="" type="radio"/>	default	double_param0	0.11
<input type="radio"/>	mode0	double_param1	9.9
<input type="radio"/>	mode1	int_param0	0
		int_param1	1
		str_param0	hoge
		str_param1	dara
		vector_param0	0.0,1.0,2.0,3.0,4.0



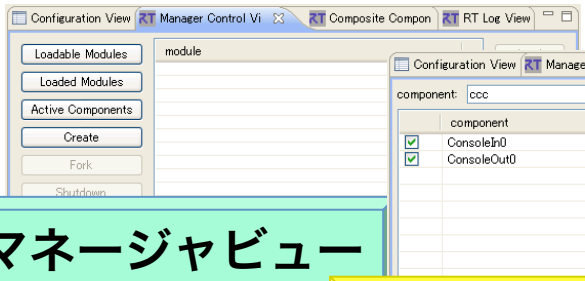
画面説明



ネームサービス
ビュー

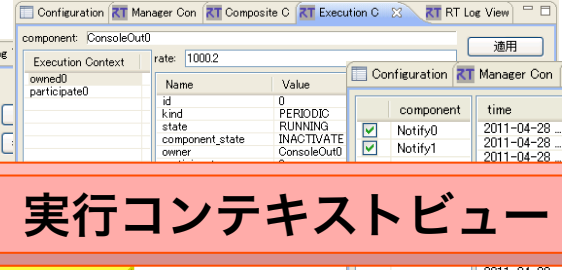
プロパティビュー

コンフィギュレーションビュー

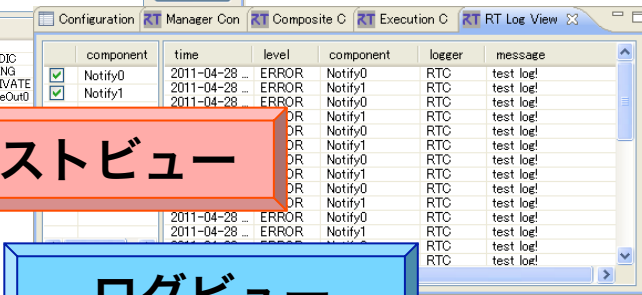


マネージャビュー

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

■ Naming Serviceの起動

■ [スタート]メニューから

[プログラム]→[OpenRTM-aist 1.1]→[tools]

→[Start C++ Naming Service]

■ ConsoleInComp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]

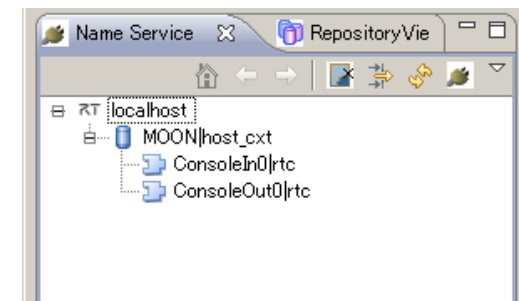
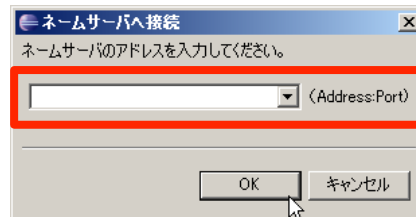
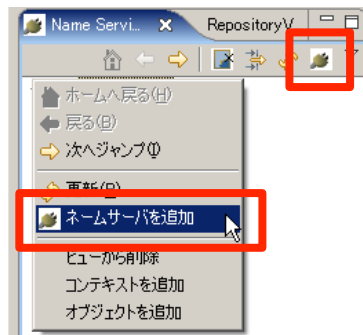
■ ConsoleOutCompの起動

■ [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]

ネームサービスへの接続

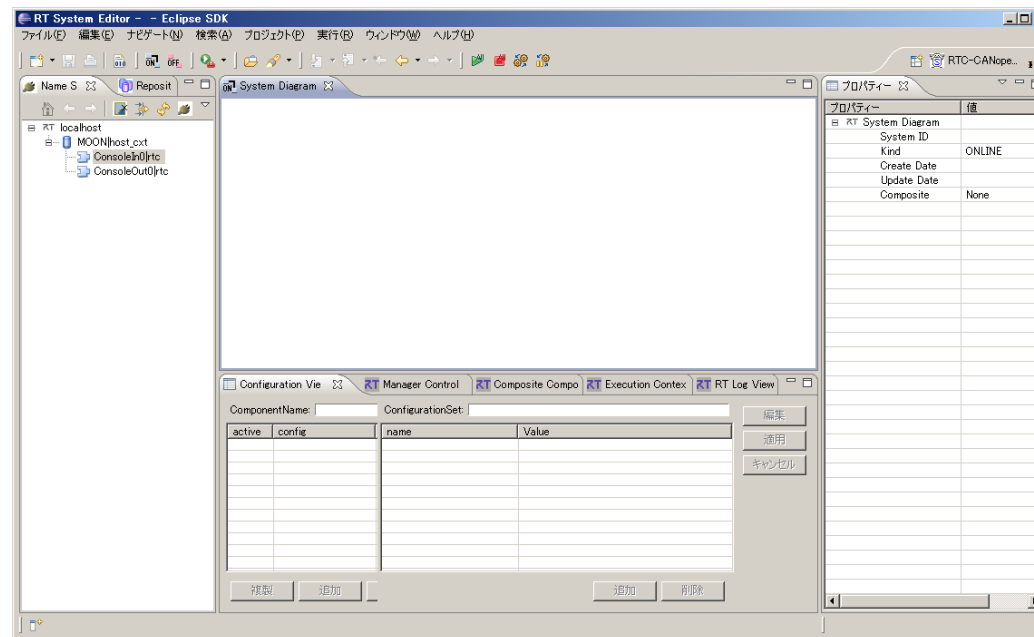
■ ネームサービスへ接続



※対象ネームサーバのアドレス，ポートを指定

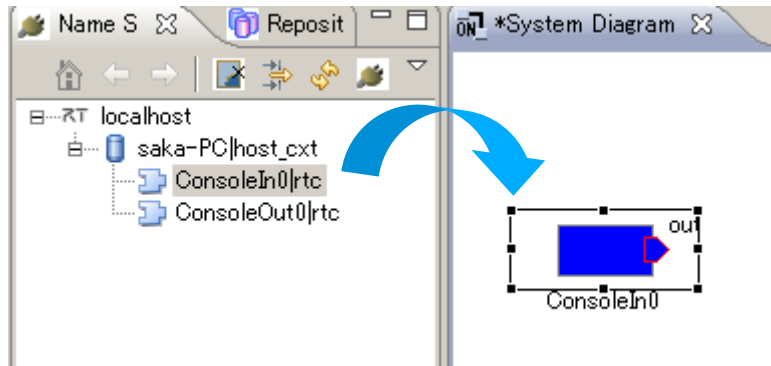
→ポート省略時のポート番号は設定画面にて設定可能

■ システムエディタの起動

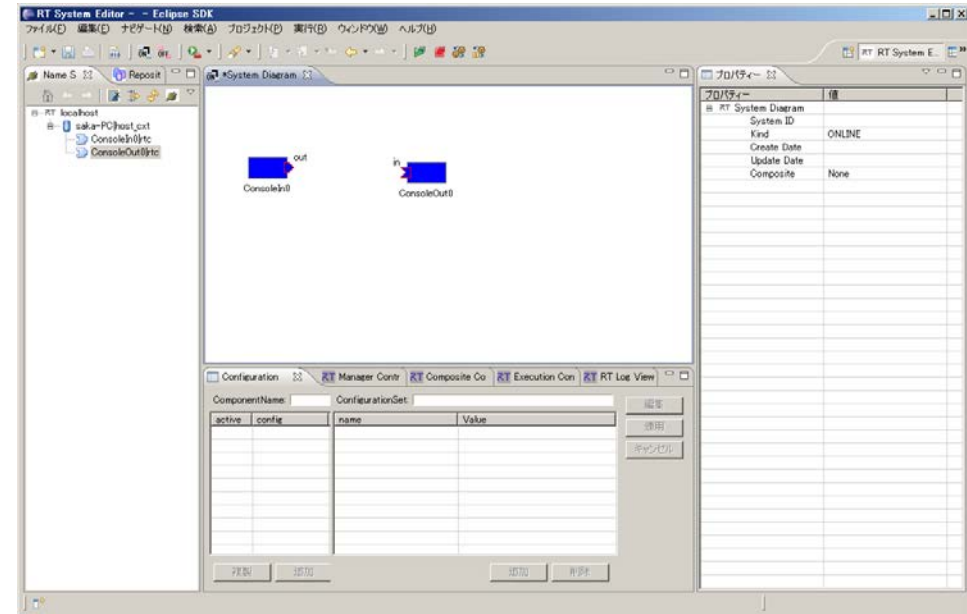


RTコンポーネントの配置方法

■ RTコンポーネントの配置

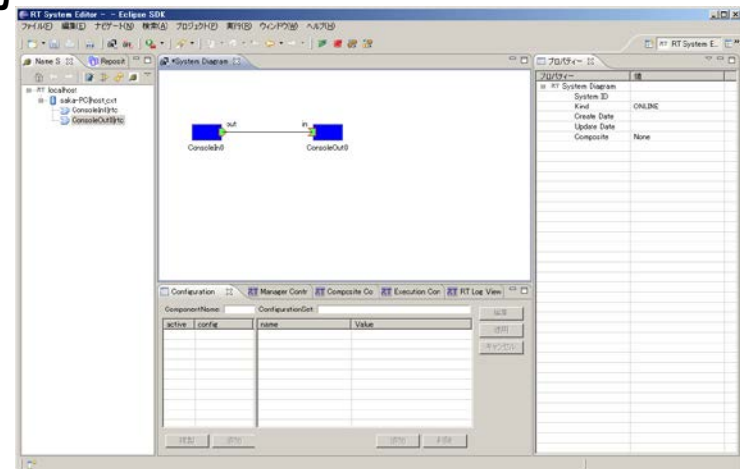
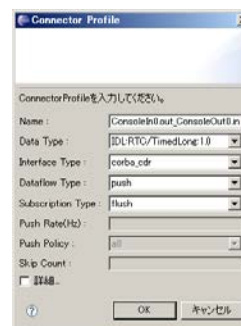
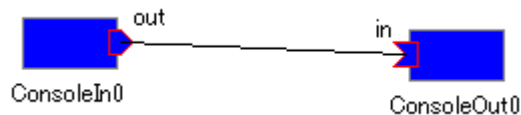


※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ



■ 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



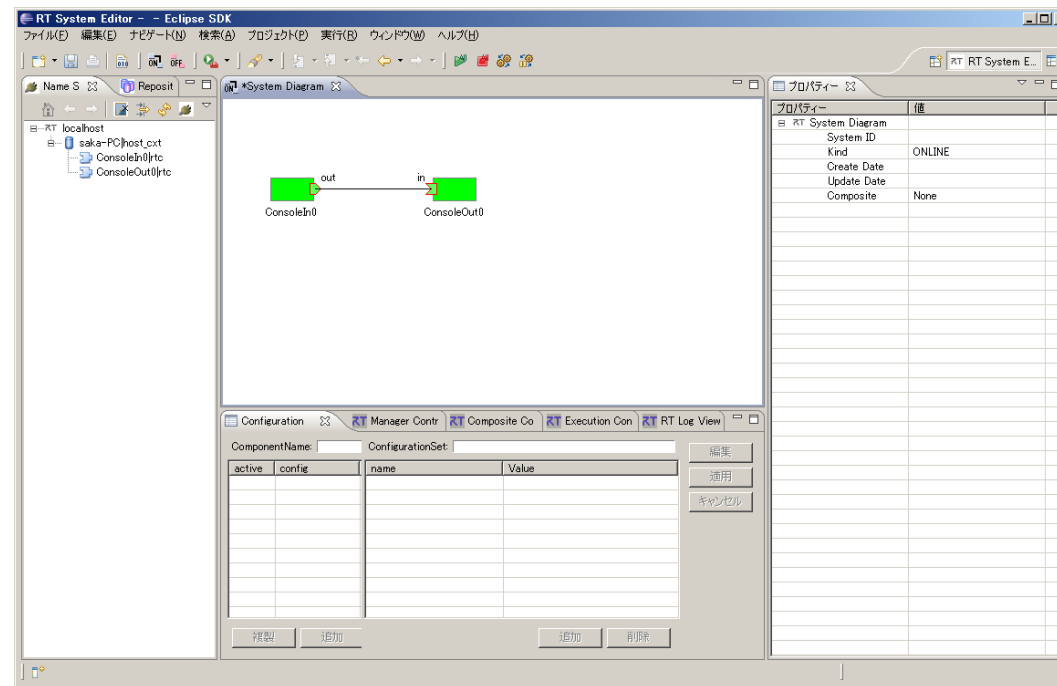
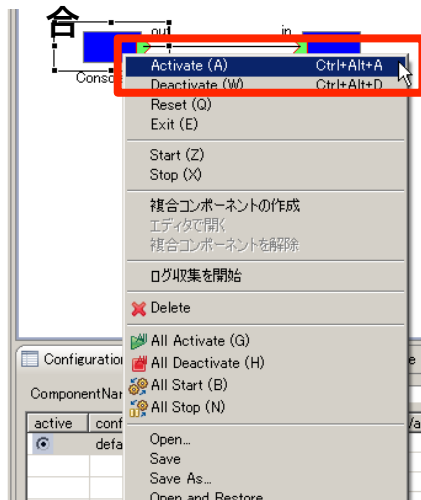
※ポートのプロパティが異なる場合など、
接続不可能なポートの場合にはアイコンが変化



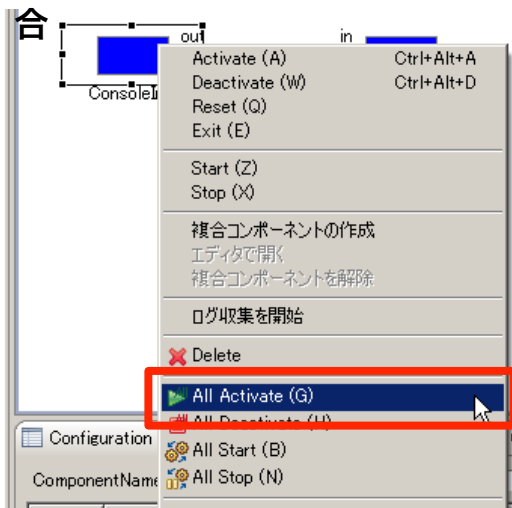
RTコンポーネントの起動

■ コンポーネントの起動

※各RTC単位で起動する場合



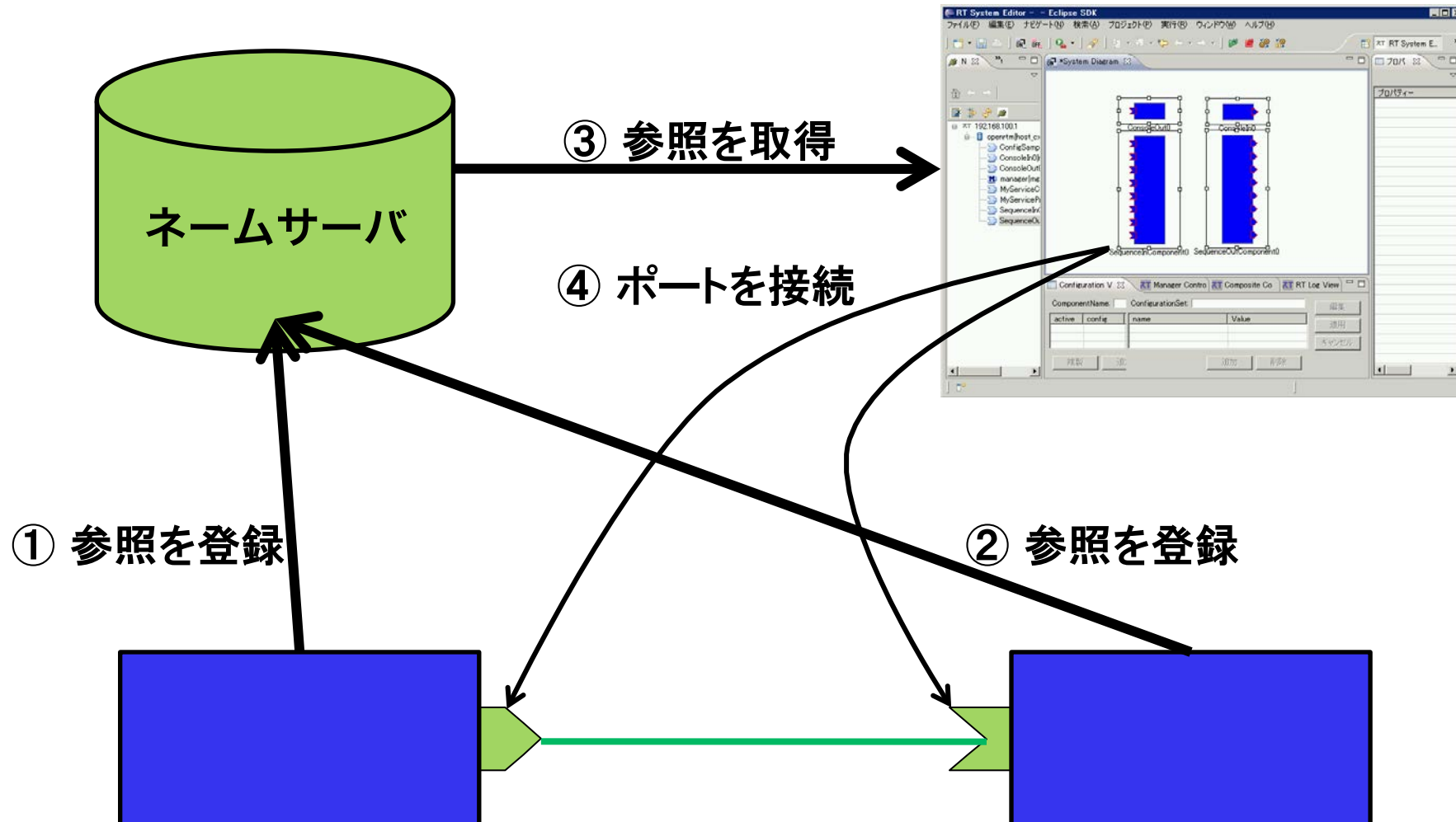
※全てのRTCを一括で起動する場合



※停止はDeactivateを実行

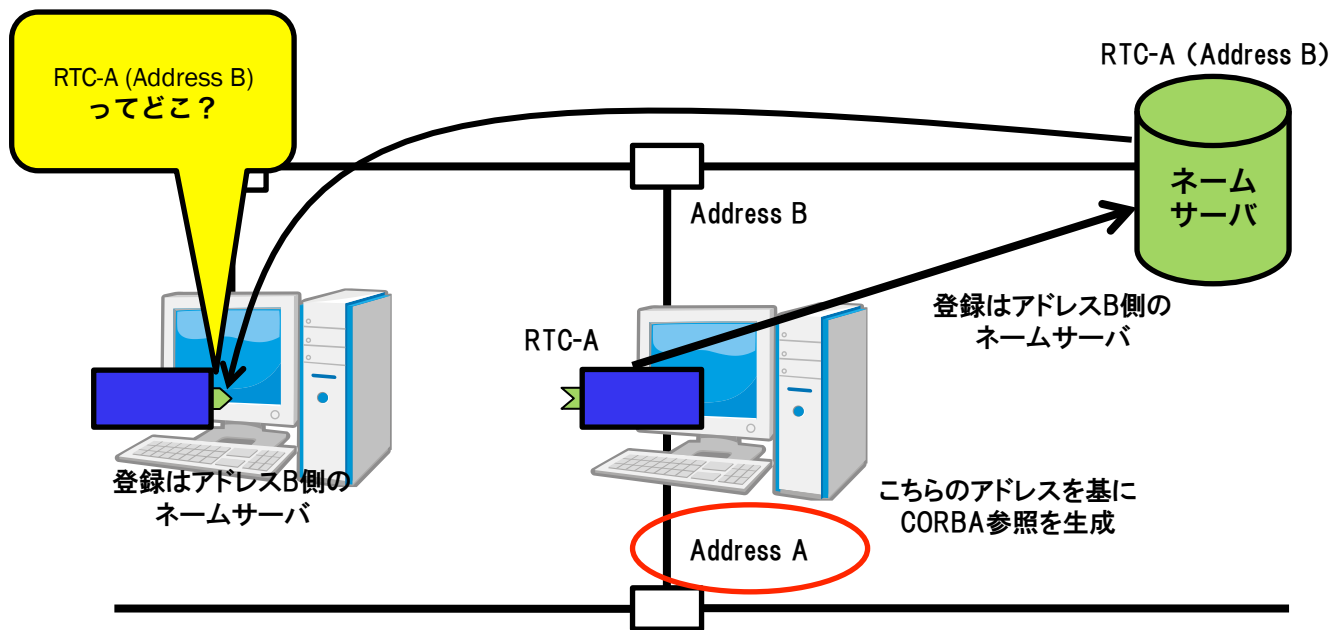
※RTC間の接続を切る場合には接続線をDelete
もしくは、右クリックメニューから「Delete」を選択

RTコンポーネントとネームサーバ



RTシステム構築時の注意点

■ 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

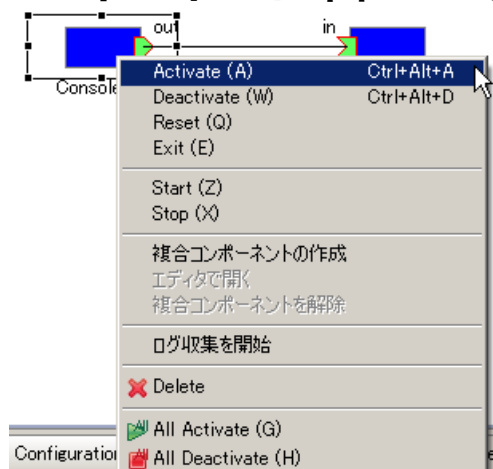
■ `naming.formats`: SimpleComponent/%n.rtc

■ `corba.endpoints`:192.168.0.12:

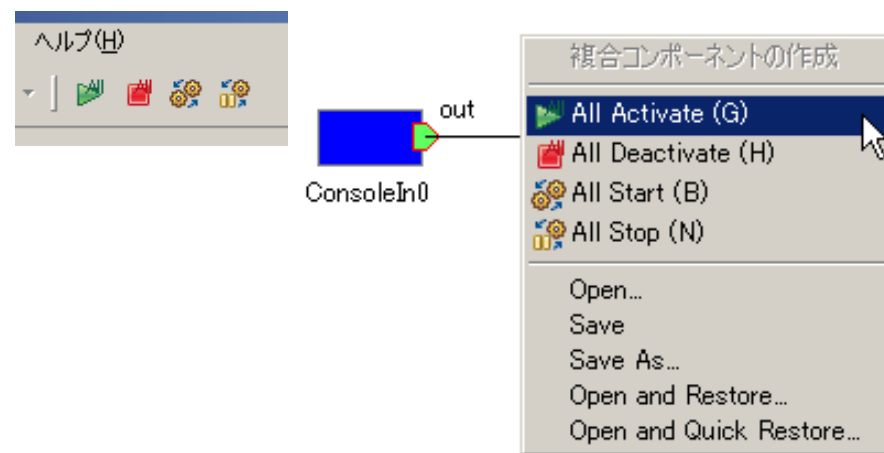
RTコンポーネントの動作

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

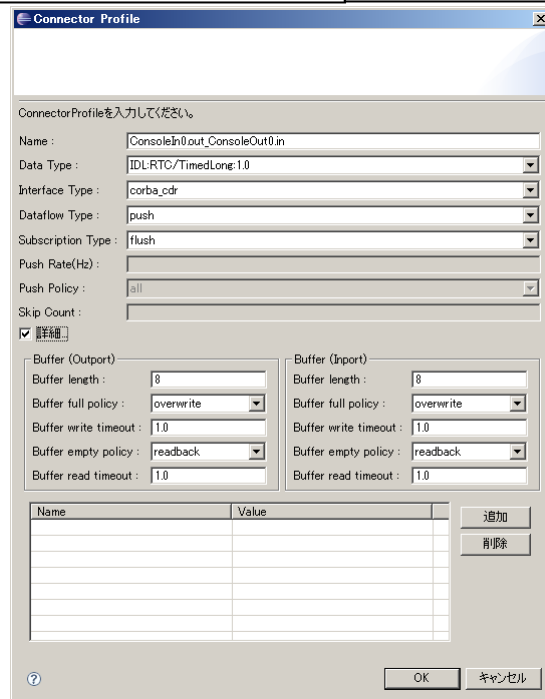


項目	設定内容
Name	接続の名称
DataType	ポート間で送受信するデータの型. ex)TimedOctet, TimedShortなど
InterfaceType	データを送受信するポートの型. ex)corba_cdrなど
DataFlowType	データの送信方法. ex)push, pullなど
SubscriptionType	データ送信タイミング. 送信方法がPush の場合有効. New, Periodic, Flushから選択
Push Rate	データ送信周期(単位: Hz). SubscriptionTypeがPeriodic の場合のみ有効
Push Policy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

※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

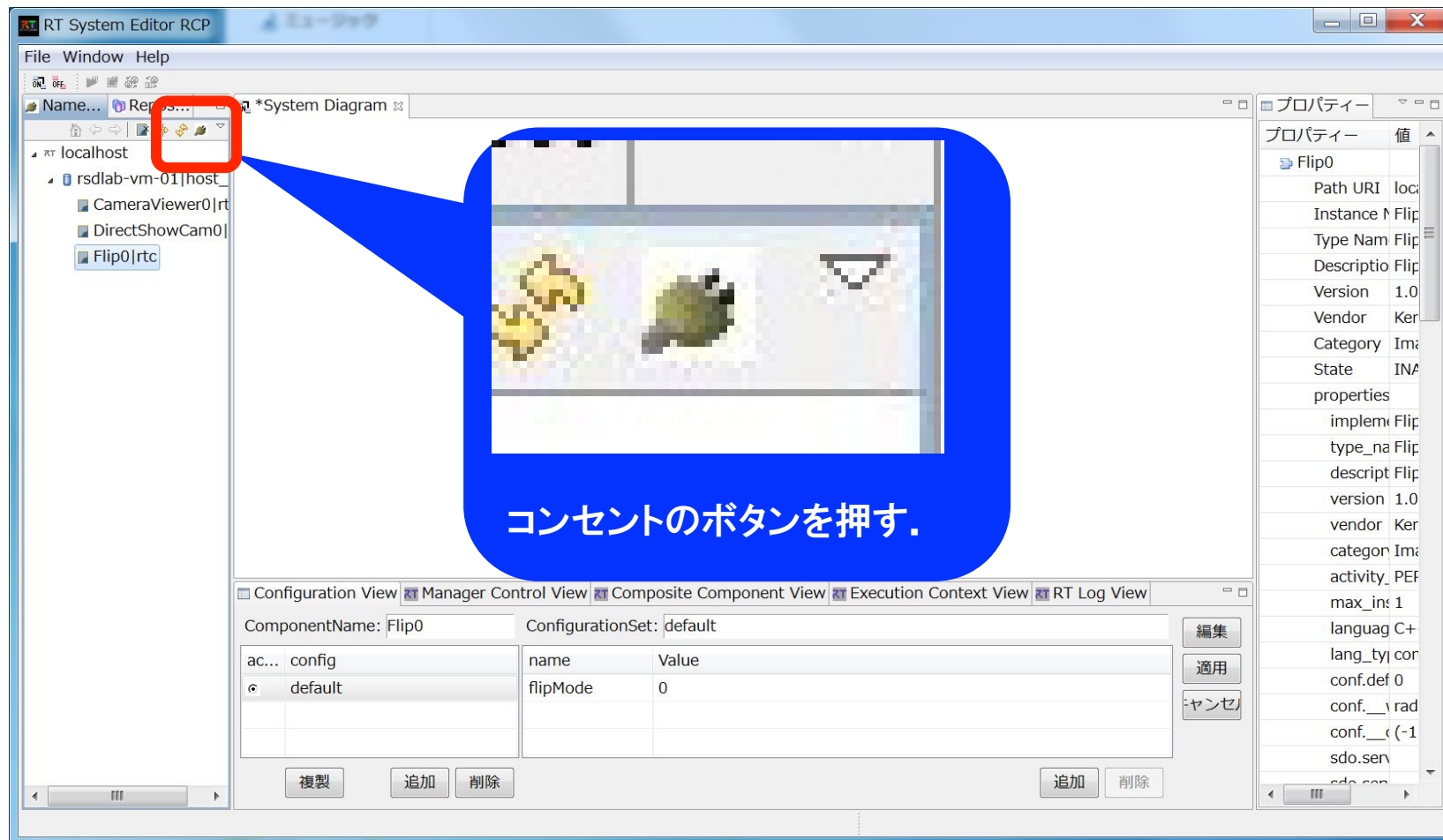
Name	Value

追加
削除

OK キャンセル

ネットワーク上のPC間でのシステム構築

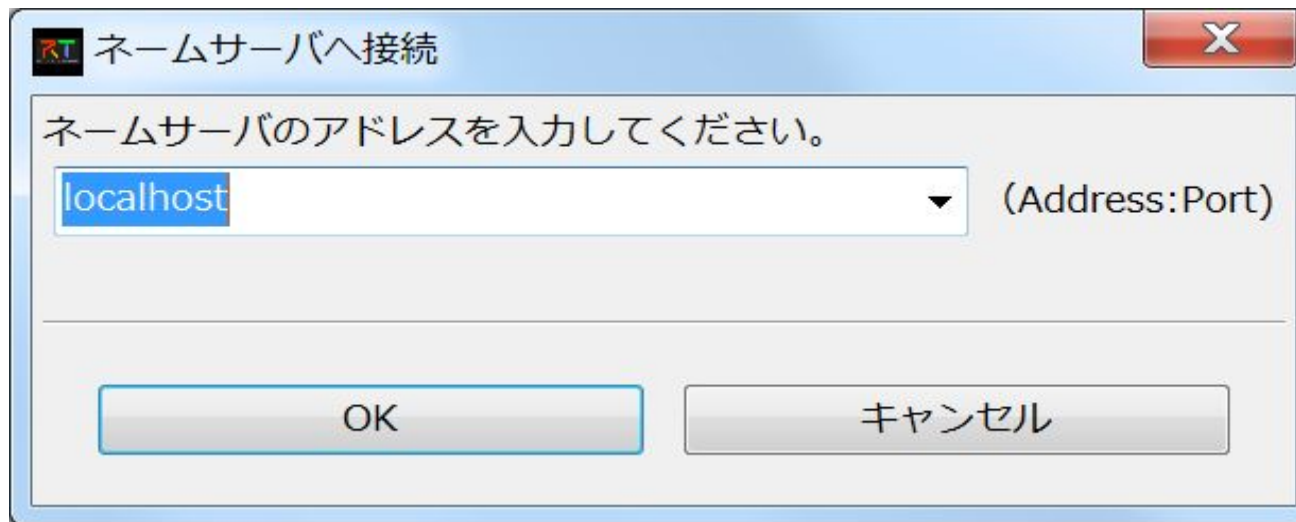
アクセス可能なネットワーク上に存在する別のPCで動作する
ネーミングサービスにアクセス



ネットワーク上のPC間でのシステム構築



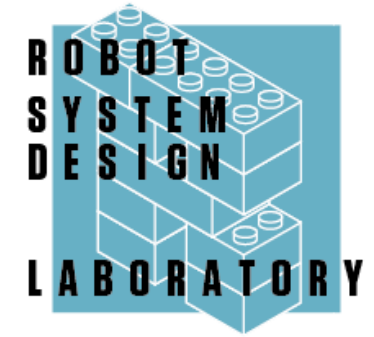
参照したいネーミングサービスが起動しているPCのIPアドレスとポートを入力する。



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))

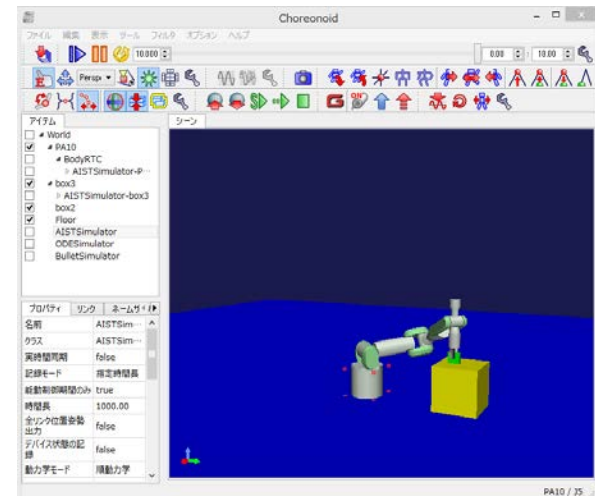
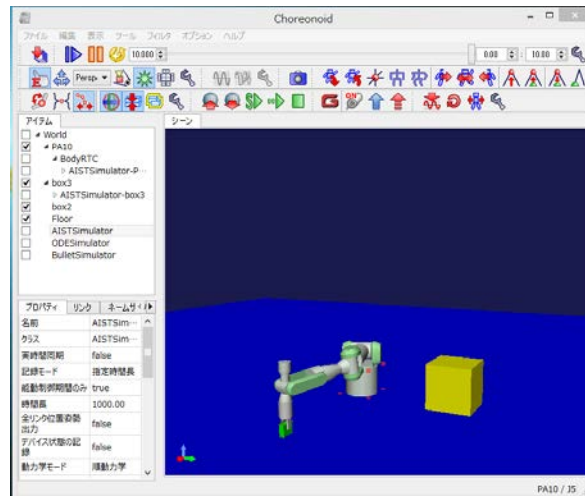
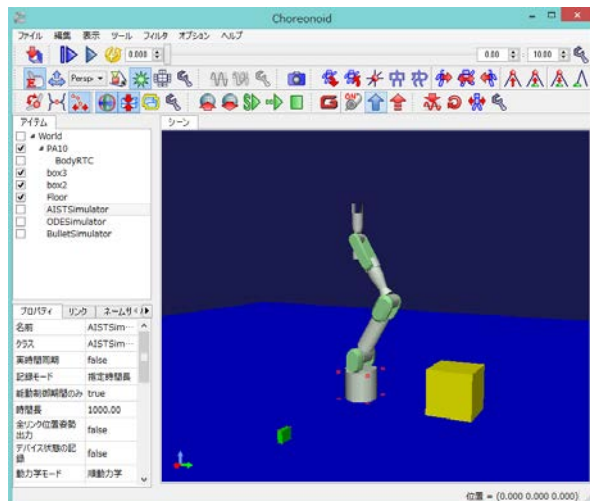


コンポーネント開発ツール RTC Builderについて



想定するコンポーネントのモデル

- ロボットアームの制御用RTコンポーネント
 - 手先位置を任意の位置に移動するためのRTC
 - アーム先端に取り付けられたグリッパの開閉が可能なRTコンポーネント



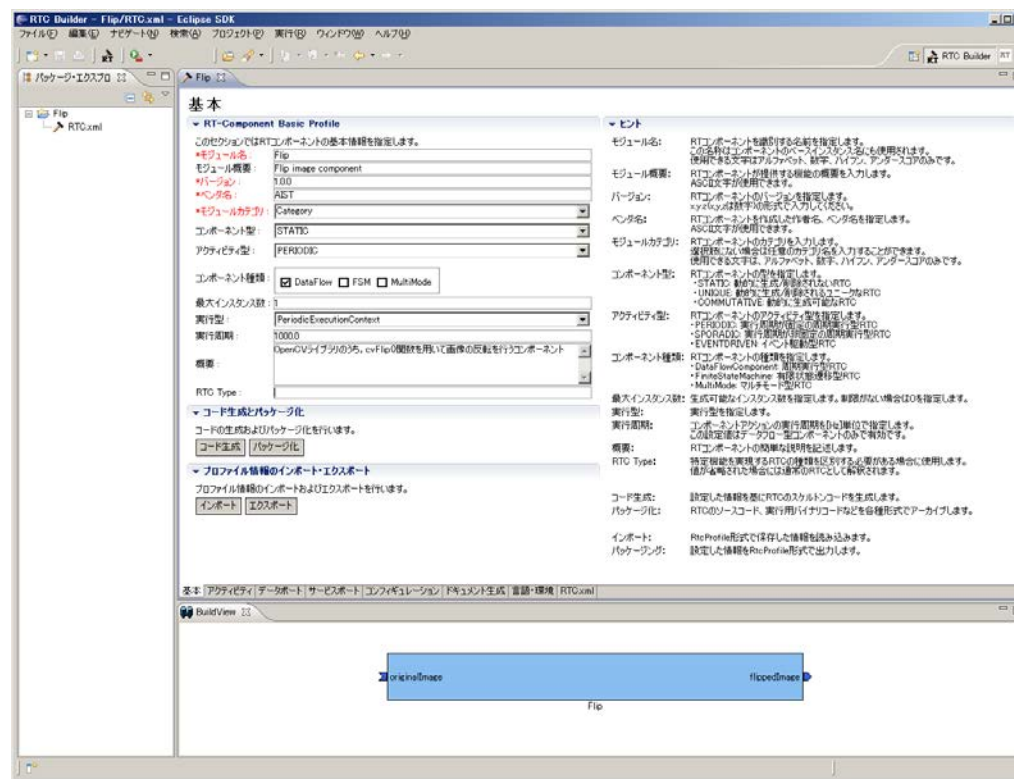
RTCBuilderの概要

■ RTCBuilderとは？

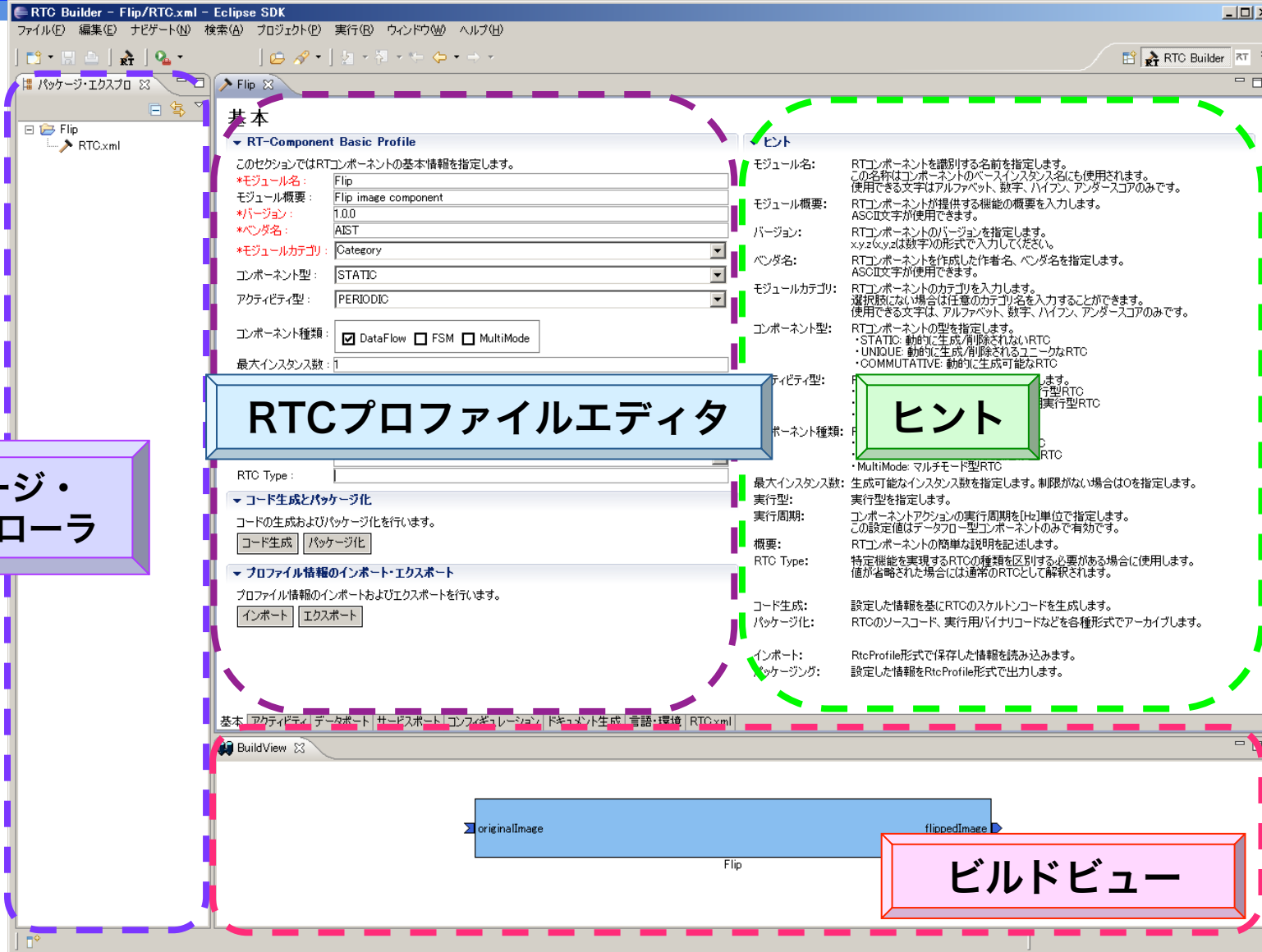
- コンポーネントのプロファイル情報を入力し、ソースコード等の雛形を生成するツール
- 開発言語用プラグインを追加することにより、各言語向けRTCの雛形を生成することが可能

- 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



RTCBuilderの外観



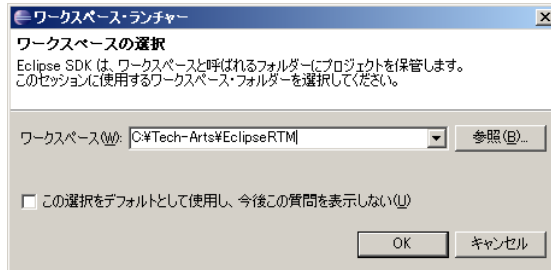
The screenshot shows the Eclipse SDK interface for RTC Builder. The main window is titled "RTC Builder - Flip/RTC.xml - Eclipse SDK". The interface is divided into several panes:

- Package Explorer:** Located on the left, showing a project named "Flip" containing an "RTC.xml" file. A callout box labeled "パッケージ・エクスプローラ" points to this pane.
- RTC Profile Editor:** The central pane, titled "基本" (Basic), showing the "RT-Component Basic Profile" for the "Flip" component. It includes fields for Module Name (Flip), Summary (Flip image component), Version (1.0.0), Vendor Name (AIST), Component Type (STATIC), and Activity Type (PERIODIC). A callout box labeled "RTCプロフィールエディタ" points to this pane.
- Hint:** A pane on the right titled "ヒント" (Hint) providing detailed instructions for each field in the profile editor. A callout box labeled "ヒント" points to this pane.
- Build View:** Located at the bottom, showing a visual representation of the component. It displays two boxes: "originalImage" and "flippedImage", with an arrow indicating the transformation. A callout box labeled "ビルドビュー" points to this pane.

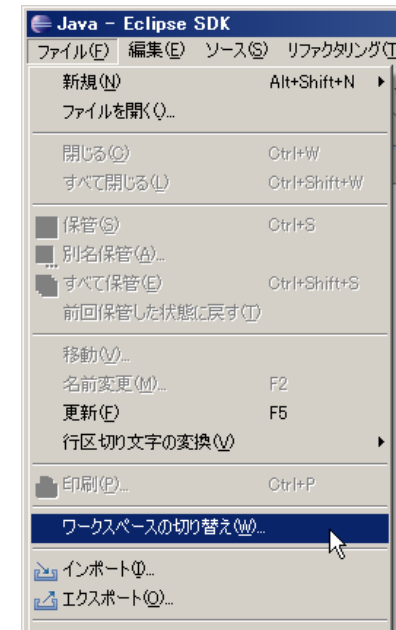
RTCBuilderの起動

- 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

初回起動の場合

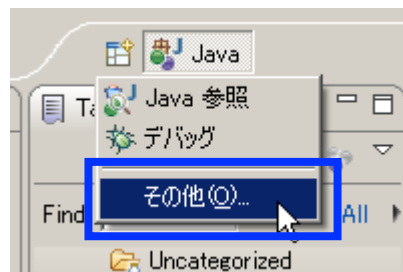
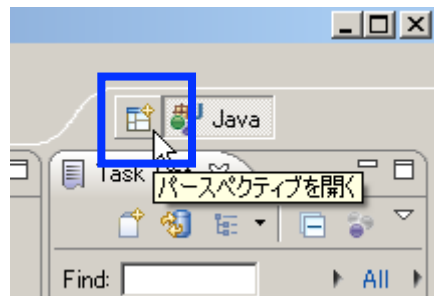
- 初期画面のクローズ
 - 初回起動時のみ



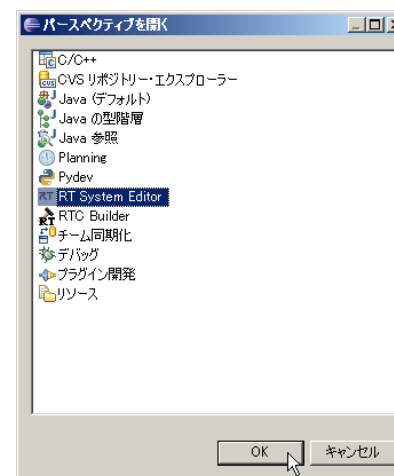
※パースペクティブ
Eclipse上でツールの構成を管理する単位
メニュー， ツールバー， エディタ， ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

■ 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

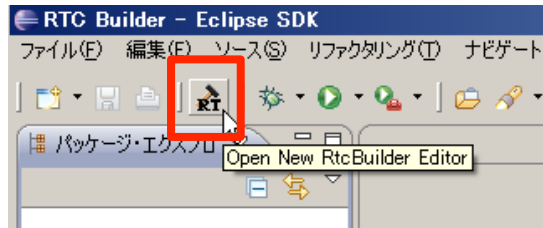


②一覧画面から対象ツールを選択

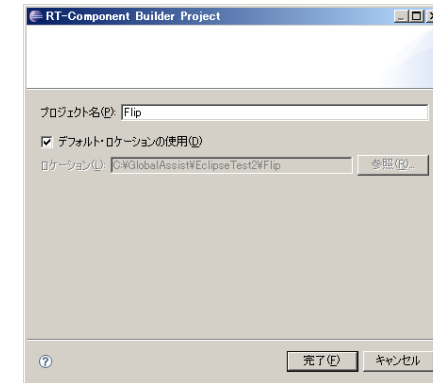


プロジェクト作成/エディタ起動

① ツールバー内のアイコンをクリック

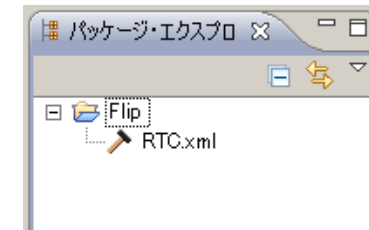


② 「プロジェクト名」欄に入力し、「終了」



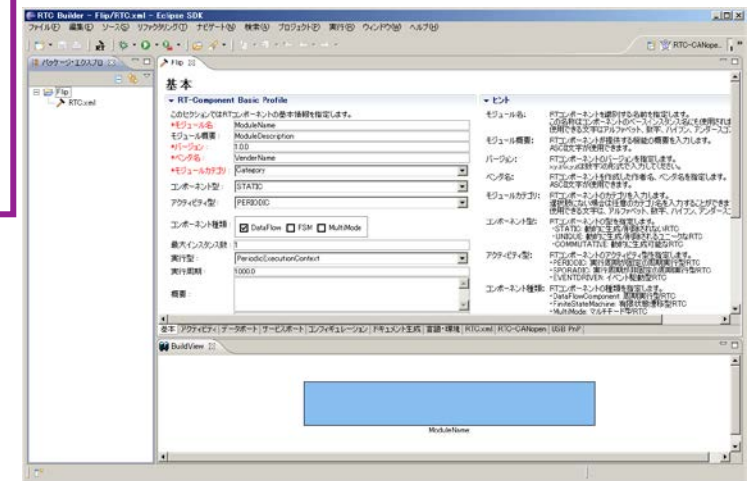
※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し、「次へ」

※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： PA10Sample



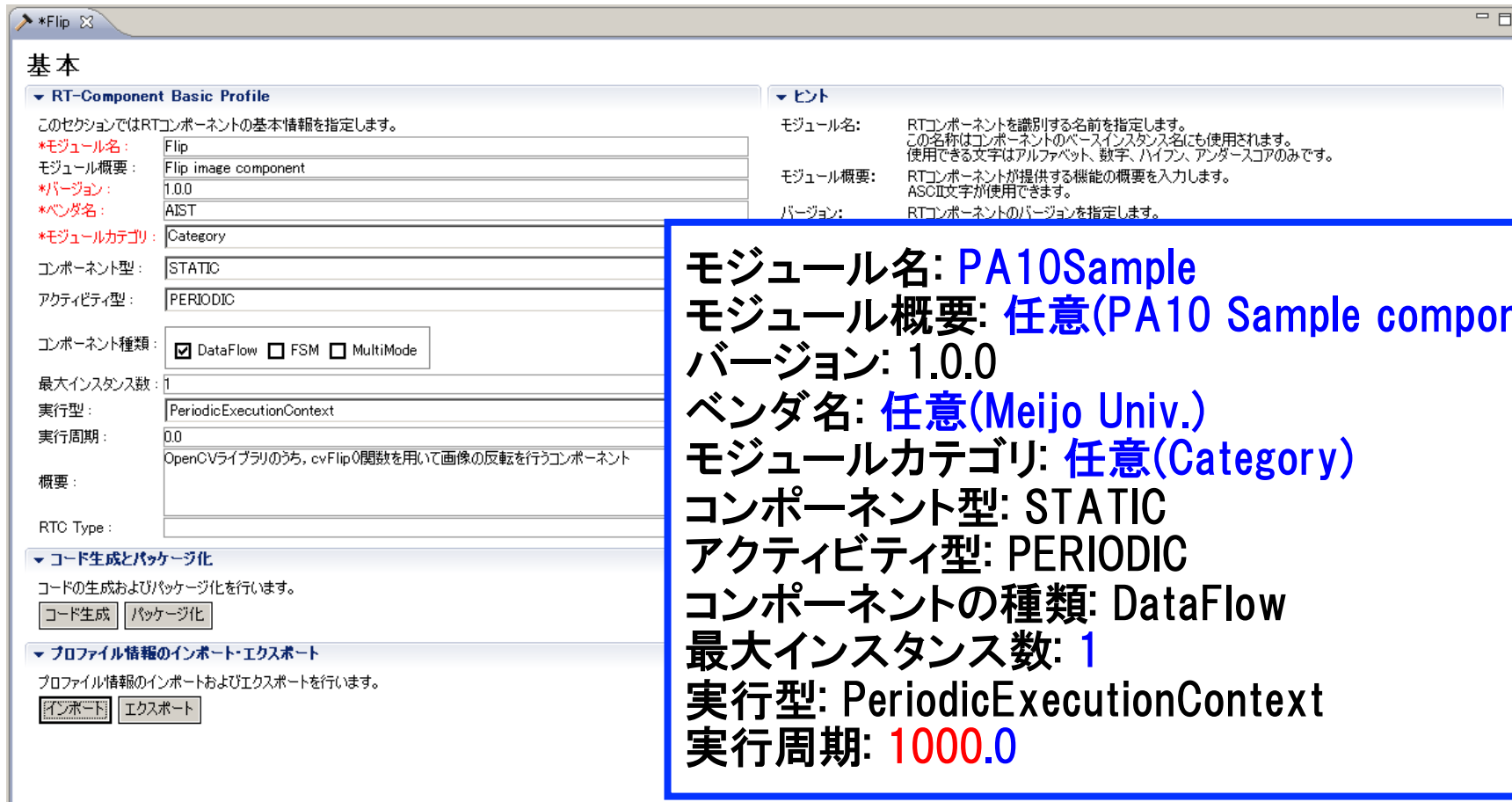
RTCプロフィールエディタ



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



基本

RT-Component Basic Profile

このセクションではRTコンポーネントの基本情報を指定します。

*モジュール名: Flip
モジュール概要: Flip image component
*バージョン: 1.0.0
*ベンダ名: AIST
*モジュールカテゴリ: Category
コンポーネント型: STATIC
アクティビティ型: PERIODIC
コンポーネント種類: DataFlow FSM MultiMode
最大インスタンス数: 1
実行型: PeriodicExecutionContext
実行周期: 0.0
概要: OpenCVライブラリのうち、cvFlipの関数を用いて画像の反転を行うコンポーネント
RTC Type:

ヒント

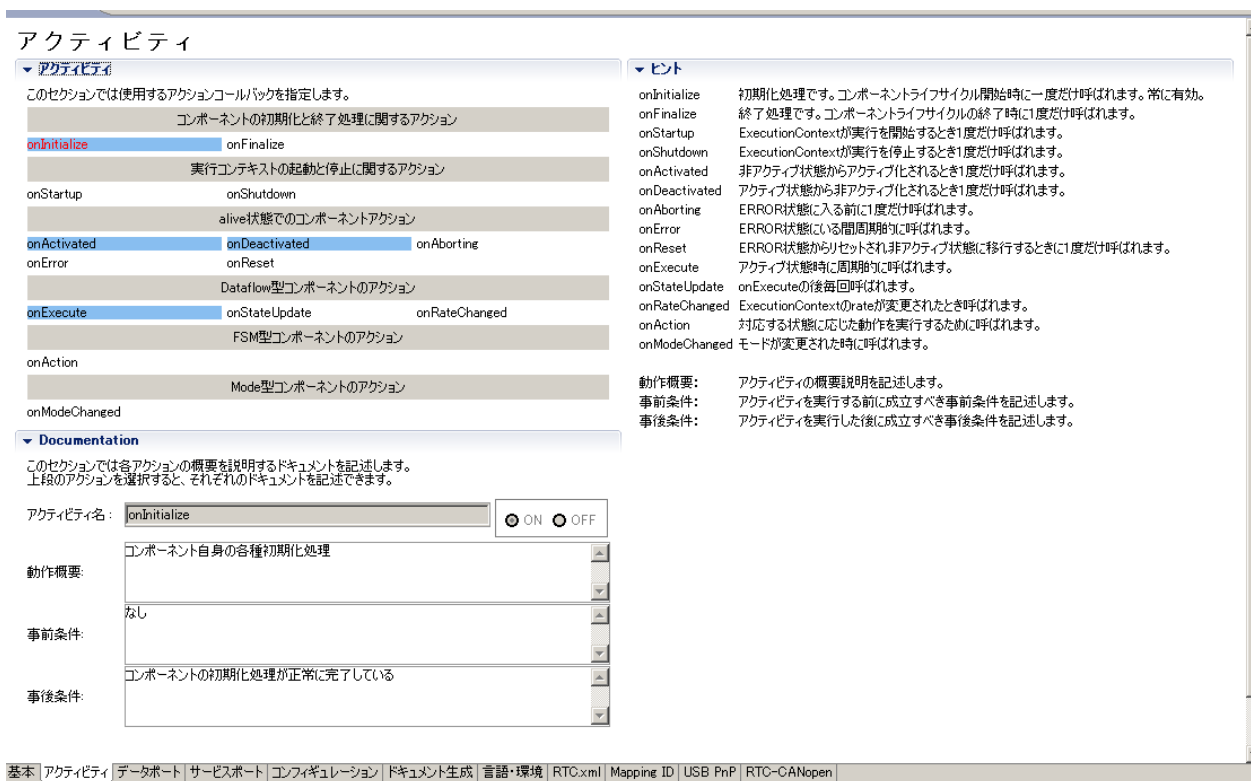
モジュール名: RTコンポーネントを識別する名前を指定します。この名称はコンポーネントのベースインスタンス名にも使用されます。使用できる文字はアルファベット、数字、ハイフン、アンダースコアのみです。
モジュール概要: RTコンポーネントが提供する機能の概要を入力します。ASCII文字が使用できます。
バージョン: RTコンポーネントのバージョンを指定します。

モジュール名: PA10Sample
モジュール概要: 任意(PA10 Sample component)
バージョン: 1.0.0
ベンダ名: 任意(Meijo Univ.)
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

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に関するアクション

onInitialize onFinalize

実行コンテキストの起動と停止に関するアクション

onStartup onShutdown

alive状態でのコンポーネントアクション

onActivated onDeactivated onAborting

onError onReset

Dataflow型コンポーネントのアクション

onExecute onStateUpdate onRateChanged

FSM型コンポーネントのアクション

onAction

Mode型コンポーネントのアクション

onModeChanged

Documentation

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

アクティビティ名: onInitialize ON OFF

動作概要: コンポーネント自身の各種初期化処理

事前条件: なし

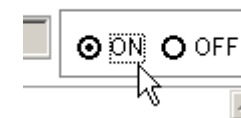
事後条件: コンポーネントの初期化処理が正常に完了している

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

① 設定対象のアクティビティを選択



② 使用/未使用を設定



以下をチェック：

onActivated
onDeactivated
onExecute

※ 現在選択中のアクティビティは、一覧画面にて赤字で表示

※ 使用(ON)が選択されているアクティビティは、一覧画面にて背景を水色で表示

※ 各アクティビティには、「動作概要」「事前条件」「事後条件」を記述可能

→ 記述した各種コメントは、生成コード内にDoxygen形式で追加される

■ 生成対象RTCに付加するDataPortの情報を設定

データポート

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

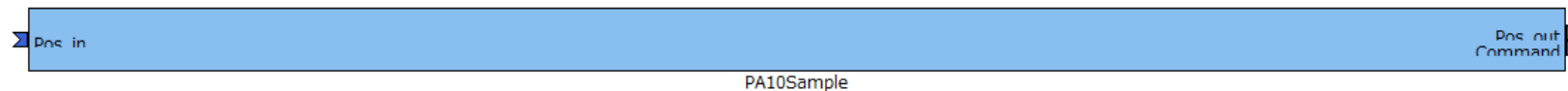
※データ型は、型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

※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: Pos_in

データ型: RTC::TimedDoubleSeq

変数名: Pos_in

表示位置: left

- OutPort1

ポート名: Command

データ型: RTC::TimedString

変数名: Command

表示位置: right

- OutPort2

ポート名: Pos_out

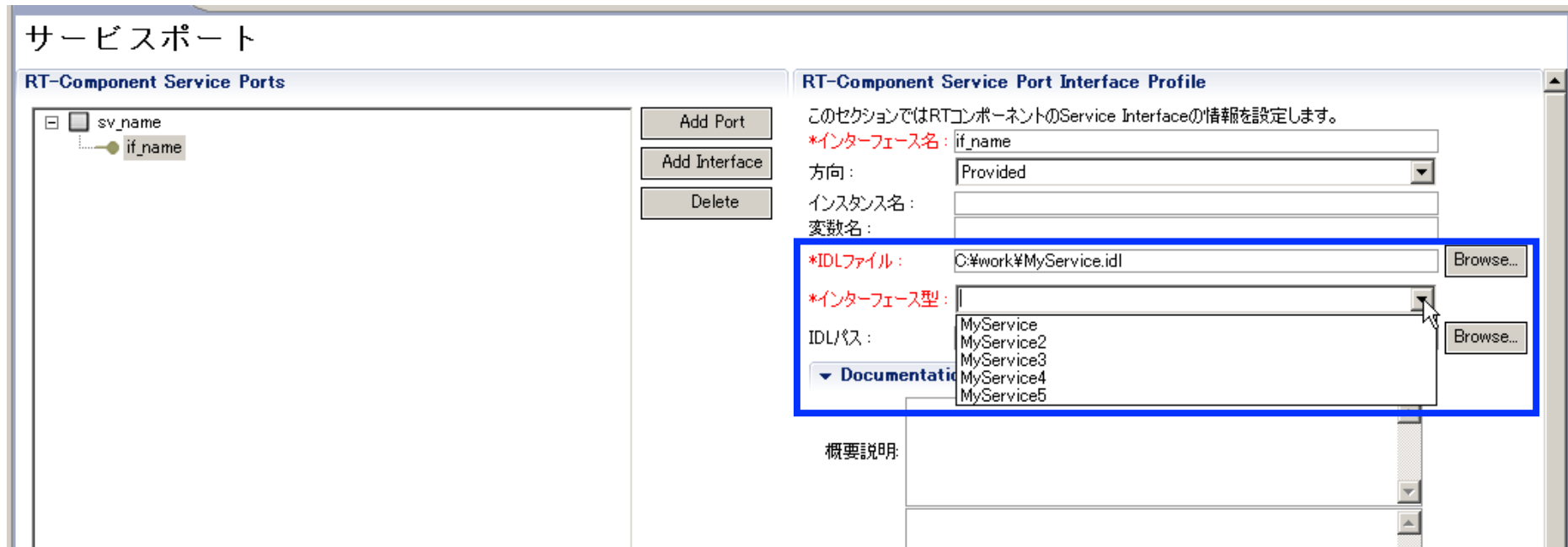
データ型: RTC::TimedDoubleSeq

変数名: Pos_out

表示位置: right

サービスポートの設定

■ 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

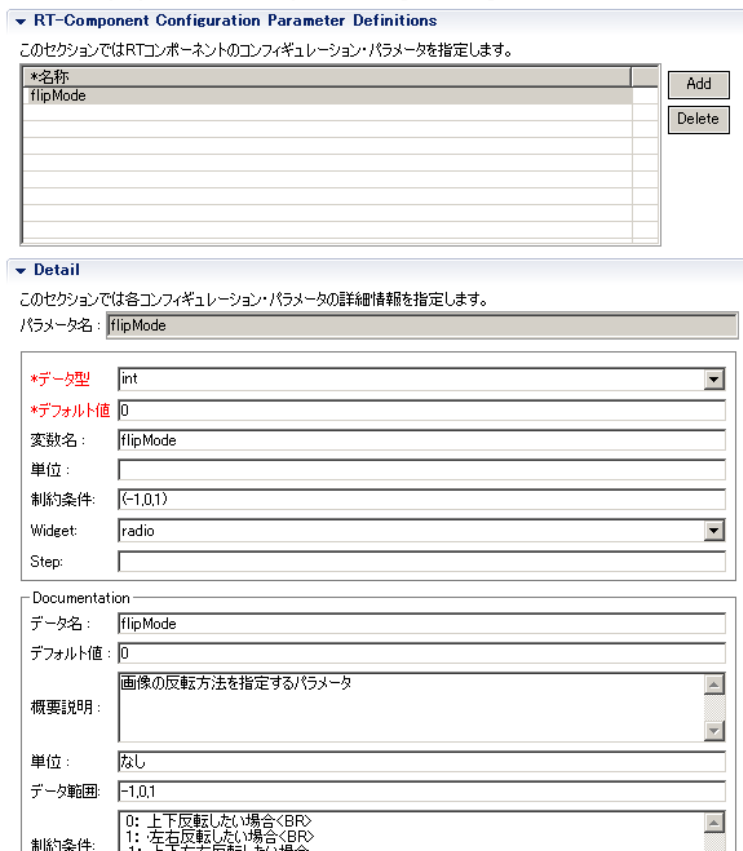
- IDLファイルを指定すると、定義されたインターフェース情報を表示

今回のサンプルでは未使用

コンフィグレーションの設定

■ 生成対象RTCで使用する設定情報を設定

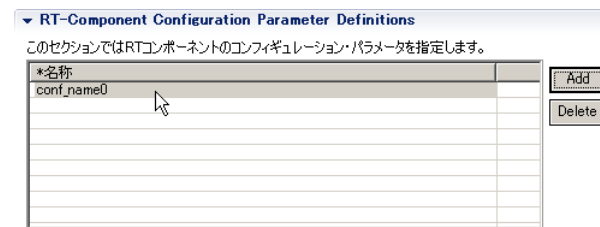
コンフィギュレーション・パラメータ



ヒント

Config. Param.: RTコン
コンフ
再利用
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
Step: 設定

①「Add」ボタンをクリックし、追加後、直接入力で名称設定



②詳細画面にて、型情報、変数名などを設定

今回のサンプルでは未使用

※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、プロフィール内のみ保存されます。

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

RTC Type :

▼ **コード生成とパッケージ化**

コードの生成およびパッケージ化を行います。

コード生成 **パッケージ化**

▼ **プロフィール情報のインポート・エクスポート**

プロフィール情報のインポートおよびエクスポートを行います。

インポート **エクスポート**

Generate success.

概要: RTコンポーネントの簡単な説明を言...

RTコンポーネントの簡単な説明を言...

RTコンポーネントの簡単な説明を言...

パッケージ・エクスプロー

- Flip
 - cmake_modules
 - cmake_uninstall.cmake
 - CPackWDX.cmake
 - cpack_resources
 - Description.txt
 - License.rtf
 - License.txt
 - wix.xsl.in
 - CMakeLists.txt
 - Doxyfile.in
 - Flip.conf
 - Flip.cpp
 - Flip.h
 - FlipComp.cpp
 - rtc.conf
 - RTC.xml
 - RTC.xml[20111116204517]

■ コード生成実行後、パースペクティブを自動切替

困属付けられたパースペクティブを開きますか?

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

はい(Y) **いいえ(N)**

MultiMode: マルチモード型RTC
最大インスタンス数: 生成可能なインスタンス数を指定し
実行型: 実行型を指定します。
実行周期: コンポーネントアクションの実行周期
この設定値はデータフロー型コンポ...

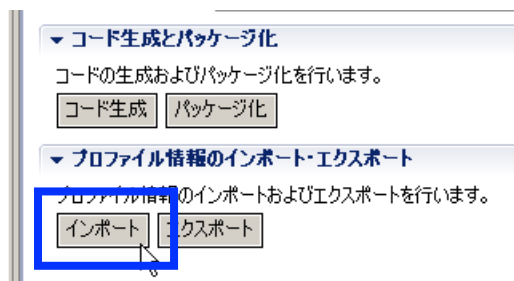
コード生成: 設定した情報を基にRTCのスケルト
パッケージ化: RTCのソースコード、実行用バイナ

※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

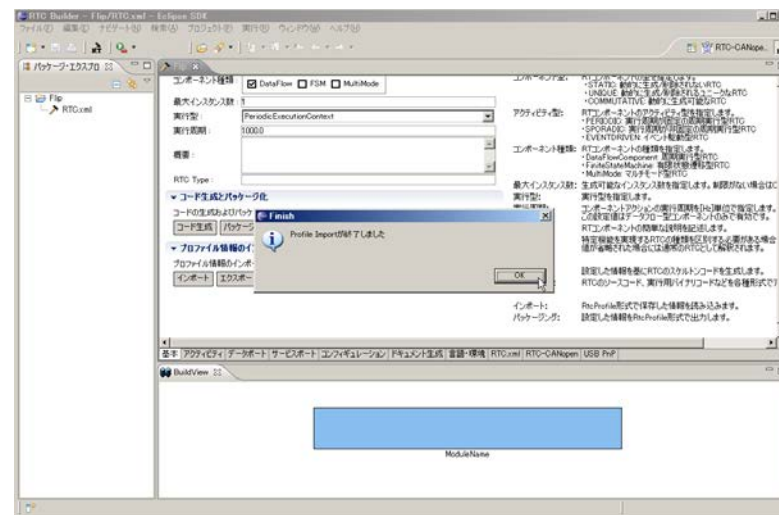
C++版RTC → CDT
Java版RTC → JDJ
(デフォルトインストール済み)
Python版 → PyDev

既存のRTCの設定を利用する場合

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

- RTC Builderで出力したファイル群そのものでは, RTCの実行ファイルの生成はできない.
- Cmakeを利用し, ソースファイルのコンパイルに必要な設定が含まれたVisual Studio用のソリューションファイルを生成する.
 - Linuxの場合はソースファイルのコンパイルに必要な設定が含まれたMakefileを生成する.
- CMakeの起動(Windows 7の場合)
 - 「スタート」->「すべてのプログラム」->「CmakeX.X」->「CMake(CMake-gui)」
- Ubuntuの場合
 - Dushホームから, CMakeと入力するとCMake-guiがでてくるので, それを利用.

Cmakeの起動画面・説明



1. ソースファイルの場所を入力

2. ソリューションファイルなどを出力する場所を入力。区別しやすいように「build」というフォルダを指定することが多い。

4. 「Generate」を押すと、ソリューションファイルなどが生成

3. 「Configure」のボタンを押すと、指定されたソースコードをコンパイルするのに必要な情報を収集する。

生成したソースファイルをCMake

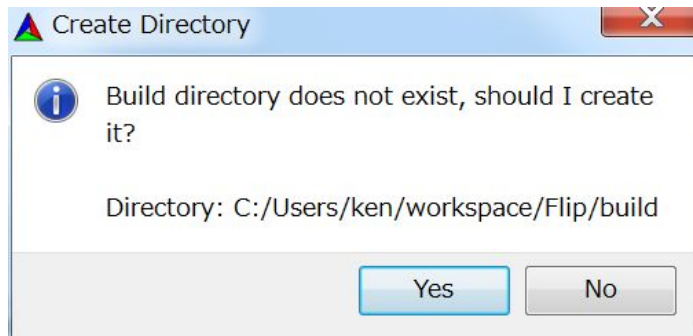


The screenshot shows the CMake GUI with the following fields and annotations:

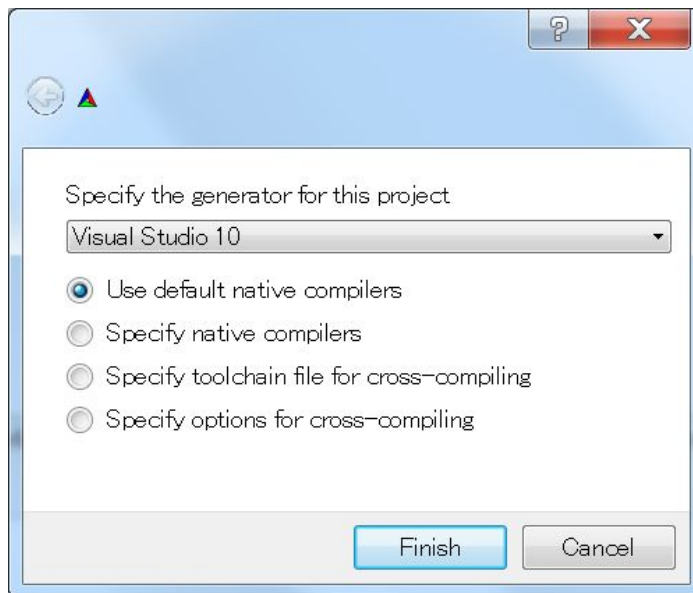
- Where is the source code:** C:/Users/ken/workspace/Flip (Annotated with a red box and a callout: "RTC Builderで生成したプロジェクトのフォルダの中にbuildというフォルダを生成して, cmakeの結果")
- Where to build the binaries:** C:/Users/ken/workspace/Flip/build (Annotated with a red box and a callout: "RTC Builderで生成したプロジェクトのフォルダを指定")
- Buttons:** "Configure" (Annotated with a red box and a callout: "上記のソースコードの場所などの指定が終わったら, 「Configure」を押す") and "Generate".

Press Configure to update and display new values in red, then press Generate to generate selected build files.

生成したソースをCMake



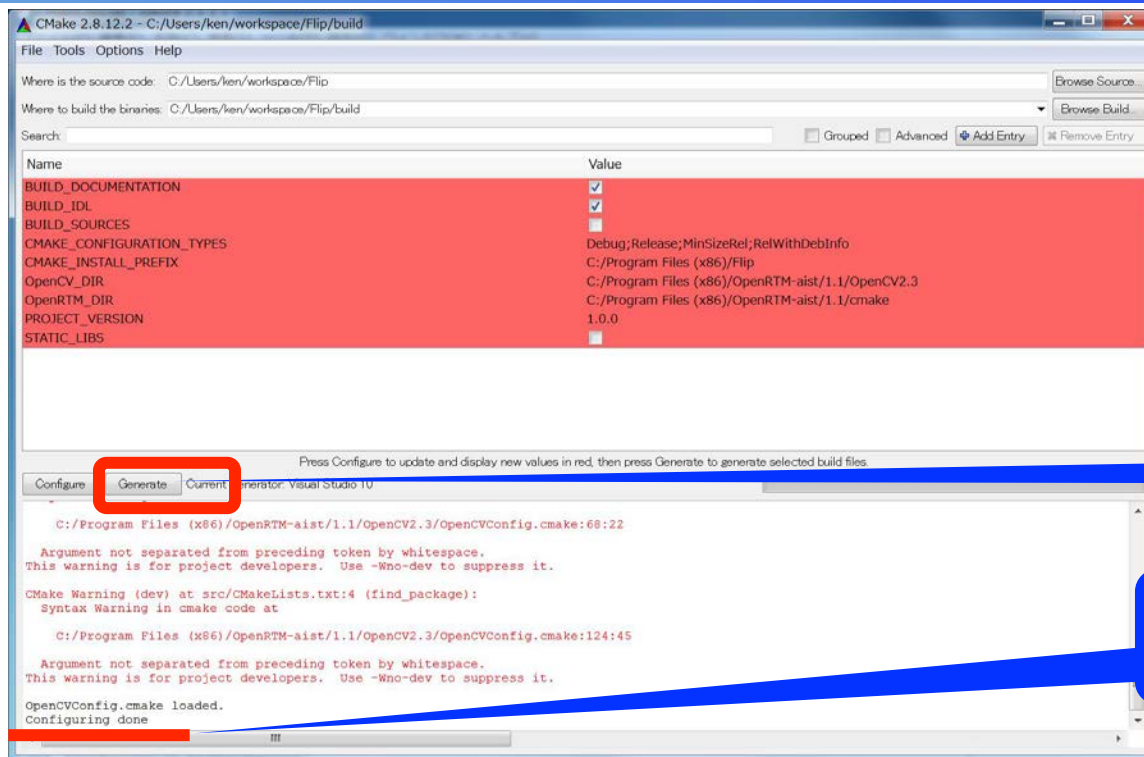
出力先に指定したbuildのフォルダがない場合、生成する旨が表示される



使用するビルド環境を指定する。
Visual Studioであればそのバージョンを指定。
(Visual Studioのバージョンとの表記の違いに注意)

Visual Studio 2010 -> Visual Studio 10
Visual Studio 2012 -> Visual Studio 11
Visual Studio 2013 -> Visual Studio 12
Linuxの場合 Unix Makefiles を指定

生成したソースをCMake



2. 「Generate」をクリック

1. 「Configuration Done」と出
ていればOK

C:/Program Files (x86)/OpenRTM-aist/1.1/OpenCV2.3/OpenCVConfig.cmake:68:22
Argument not separated from preceding token by whitespace.
This warning is for project developers. Use -Wno-dev to suppress it.
CMake Warning (dev) at src/CMakeLists.txt:4 (find_package):
Syntax Warning in cmake code at
C:/Program Files (x86)/OpenRTM-aist/1.1/OpenCV2.3/OpenCVConfig.cmake:124:45
Argument not separated from preceding token by whitespace.
This warning is for project developers. Use -Wno-dev to suppress it.
OpenCVConfig.cmake loaded.
Configuring done
Generating done

3. 「Generation Done」と出ればOK

- 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>

- 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>