第3部の資料

- 設定ファイルの差し替え
  - [OpenRTM_Root]/1.0/bin 配下にある以下の設定ファイルを差し替え

    rtc.conf, configsample.conf

※デフォルト設定でインストールした場合

C:¥Program Files¥OpenRTM-aist¥1.0¥bin

RTミドルウエア講習会

日時：2011年1月21日(金)　11:00～17:00
場所：山形大学 次世代ロボットデザインセンター

RTミドルウェア講習会

11:00–12:00	**第1部：RTミドルウエアの現状と今後の展開について**
	担当： 神徳徹雄（産総研）
	概要：RTミドルウェアの現状および、今後の展望について。
13:00–13:45	**第2部：OpenRTM-aist サンプルコンポーネントの紹介とその利用法**
	担当：栗原眞二（産総研）
	概要：開発実習にて作成するRTCに関するサンプルコンポーネントについて紹介し、 RTCの便利さ，面白さを体感して頂きます。
14:00–15:00	**第3部：OpenRTM-aist開発支援ツールの紹介とその利用法**
	担当：Geoffrey Biggs（産総研）
	概要：RTコンポーネントを作成するツールRTCBuilder、およびRTシステムを設計するツール RTSystemEditer（GUI版）、rtshell（CUI版）の使い方について解説します。
15:15–17:00	**第4部：コンポーネント開発実習**
	担当：栗原眞二（産総研）
	概要：OpenRTMのインストール方法やテスト方法を解説します。OpenRTM-aistでのコンポーネント作成方法を実際に体験していただきます。RTCBuilderを使用したRTコンポーネントの 設計とRTSystemEditerでのRTシステム作成を行います。

OpenRTM-aist開発支援ツールの紹介とその利用法

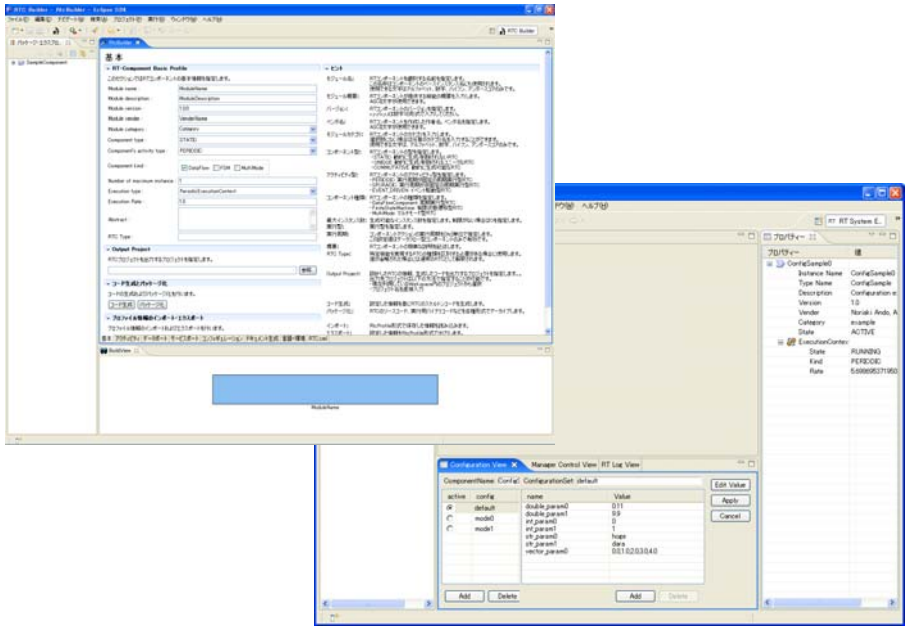
産業技術総合研究所
知能システム研究部門
Geoffrey Biggs
栗原　眞二

OpenRTM-aistの開発支援ツールについて

OpenRT Platform

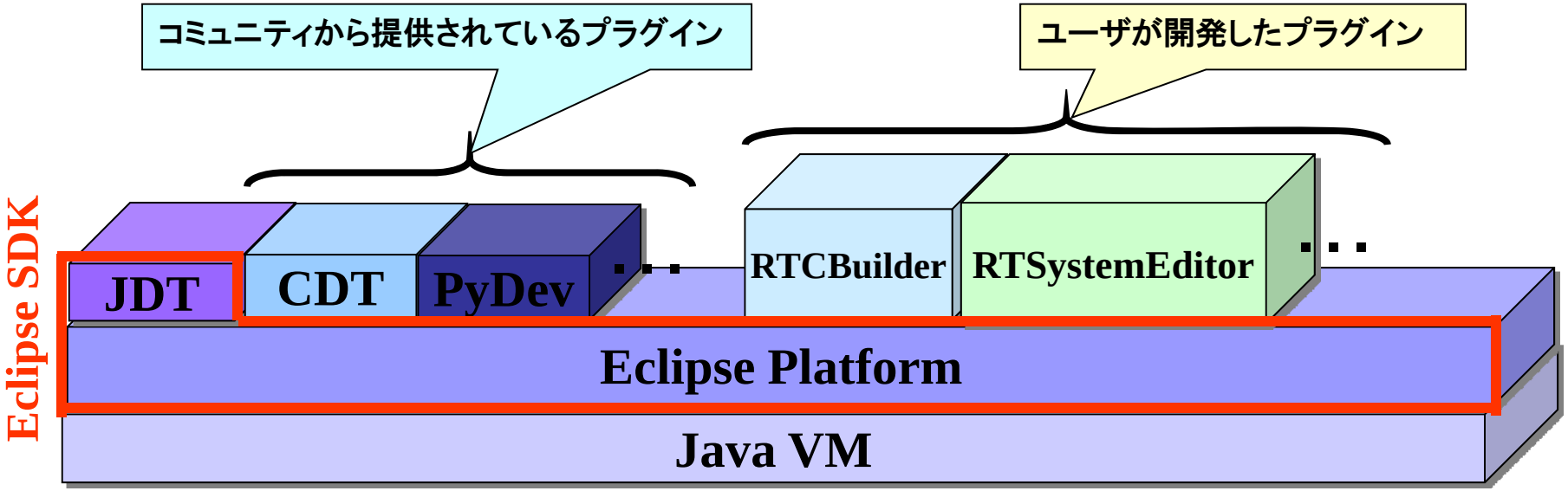
- 次世代ロボット知能ソフトウェアプラットフォーム
  - http://www.openrtp.jp/wiki/
  - システム設計，シミュレーション，動作生成，シナリオ生成などをサポート
- OpenRT Platformツール群
  - コンポーネント開発，システム開発における各開発フェーズの作業支援
  - 開発プラットフォームにEclipseを採用
- 構成
  - RTCビルダ
  - RTCデバッガ
  - RTシステムエディタ
  - ロボット設計支援ツール
  - シミュレータ
  - 動作設計ツール
  - シナリオ作成ツール

など



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



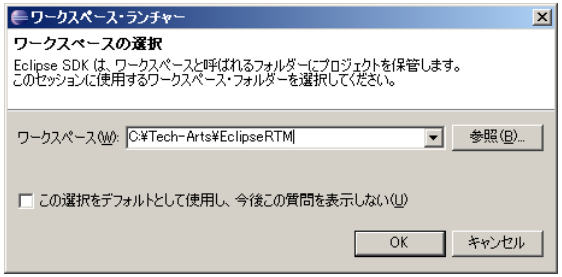
RTCBuilder,RTSystemEditorのインストール

- 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

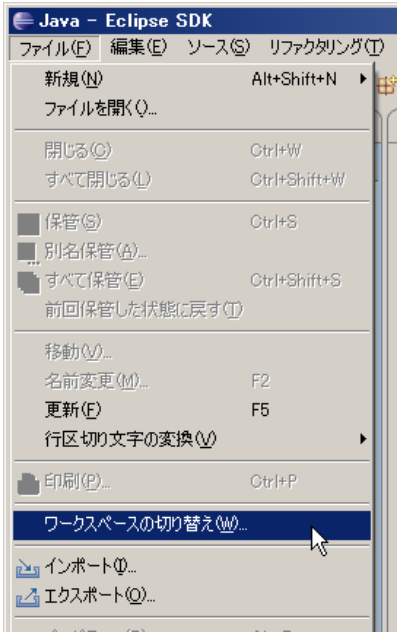
ツールの起動

- 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)



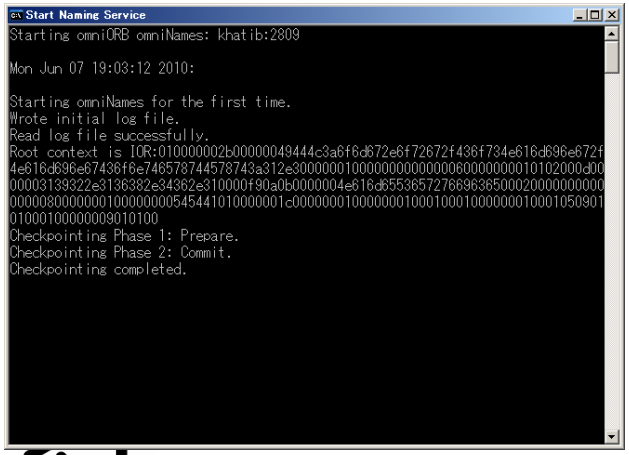
- ワークスペースの切替(通常時)

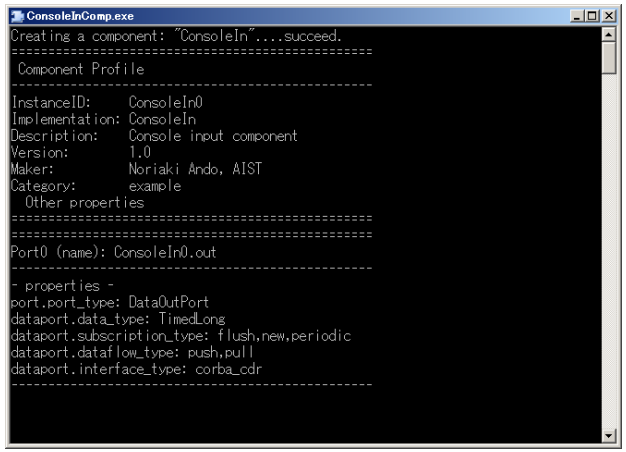


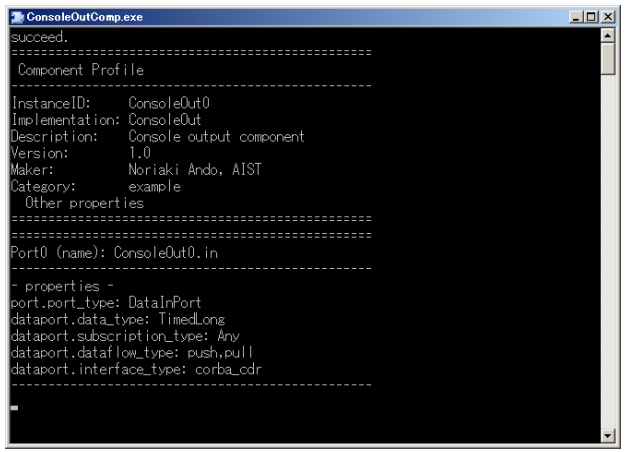
※ワークスペース
Eclipseで開発を行う際の作業領域
Eclipse上でプロジェクトやファイルを作成すると
ワークスペースとして指定したディレクトリ以下に
実際のディレクトリ，ファイルを作成する

準備

- Naming Serviceの起動
  - [スタート]メニューから
    [プログラム]→[OpenRTM-aist]→[C++]→[tools]→[Start Naming Service]
- ConsoleIn コンポーネント/ConsoleOut コンポーネントの起動
  - [スタート]メニューから
    [プログラム] →[OpenRTM-aist]→[C++]
    →[components]→[examples] →[ConsoleInComp.exe]
    [プログラム] →[OpenRTM-aist]→[C++]
    →[components]→[examples] →[ConsoleOutComp.exe]

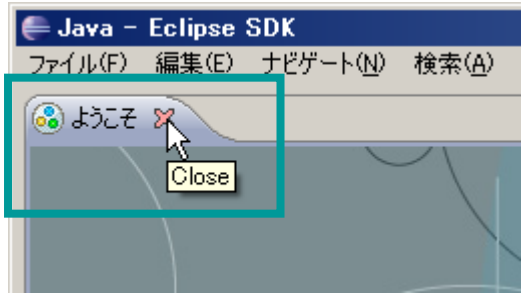






準備

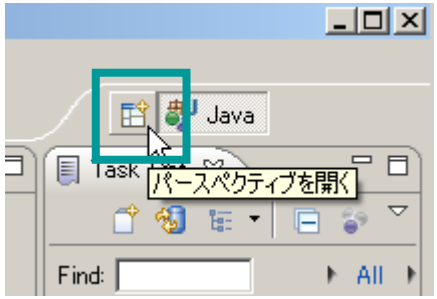
- 初期画面のクローズ
  - 初回起動時のみ

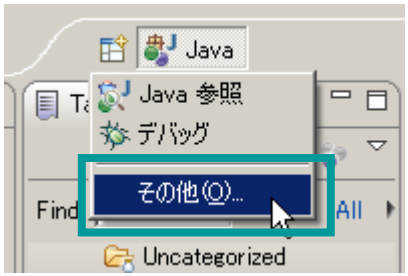


※パースペクティブ
Eclipse上でツールの構成を管理する単位
メニュー，ツールバー，エディタ，ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

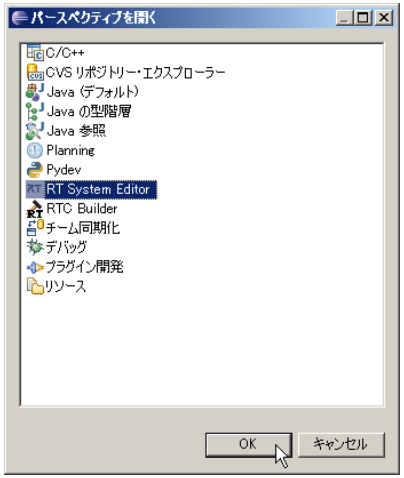
- パースペクティブの切り替え

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





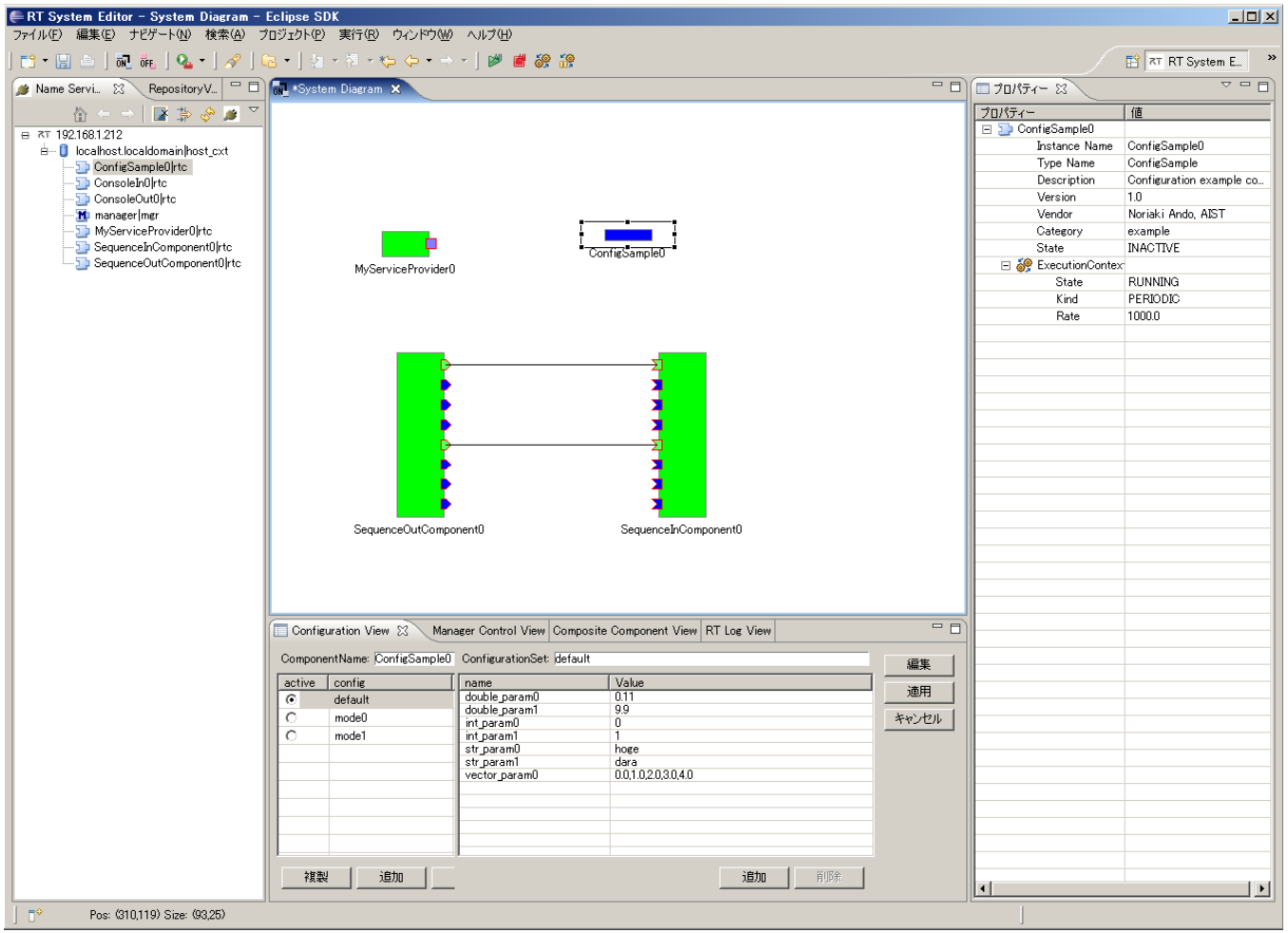
②一覧画面から対象ツールを選択



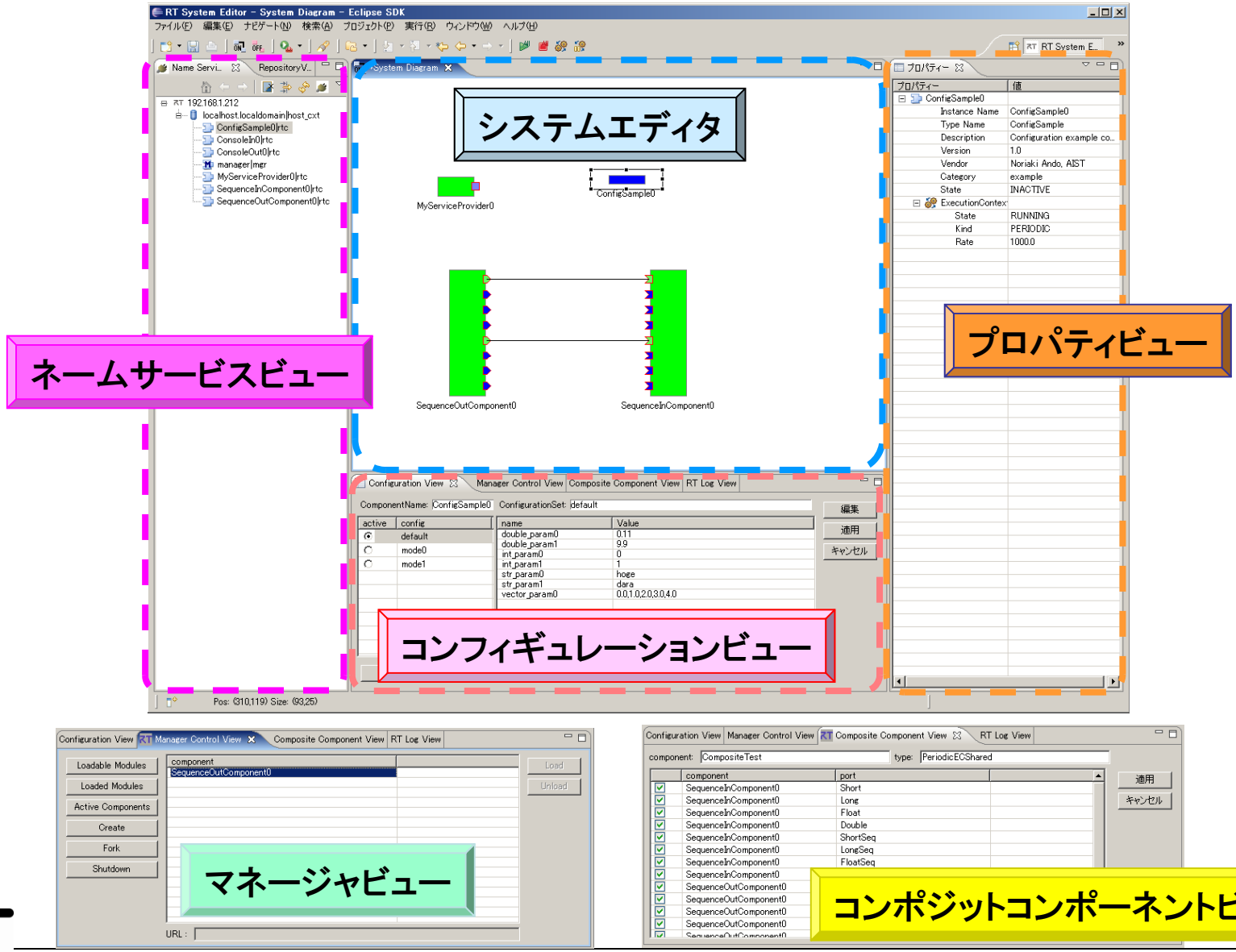
RTSystemEditorについて

RTSystemEditor概要

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

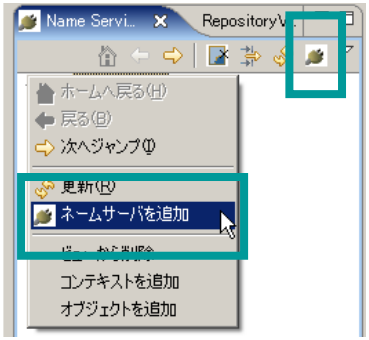


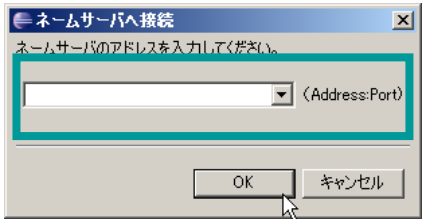
画面構成


RT System Editor - System Diagram - Eclipse SDK
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
マネージャビュー
コンポジットコンポーネントビュー

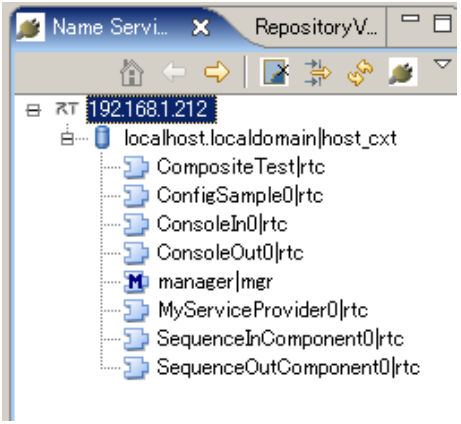
RTシステム構築の基本操作

ネームサービスへ接続

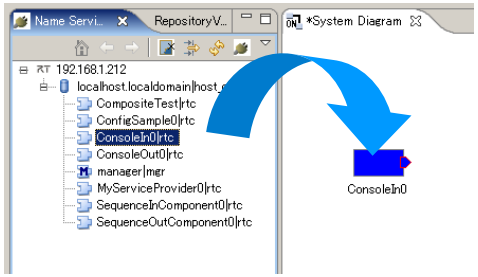




※対象ネームサーバのアドレス，ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能



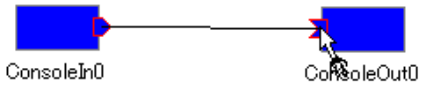
RTコンポーネントの配置



※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

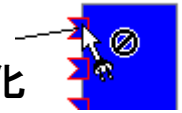
ポートの接続

①接続元のポートから接続先のポートまでドラッグ



②接続プロファイルを入力

※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



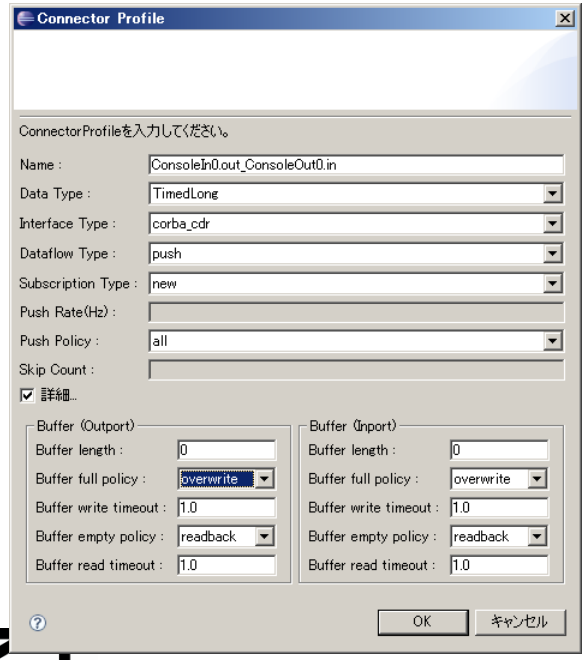
接続プロファイル(DataPort)について

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

- SubscriptionType
  - New :バッファ内に新規データが格納されたタイミングで送信
  - Periodic :一定周期で定期的にデータを送信
  - Flush :バッファを介さず即座に同期的に送信
- Push Policy
  - all :バッファ内のデータを一括送信
  - fifo :バッファ内のデータをFIFOで1個ずつ送信
  - skip :バッファ内のデータを間引いて送信
  - new :バッファ内のデータの最新値を送信(古い値は捨てられる)

接続プロファイル(DataPort)について

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



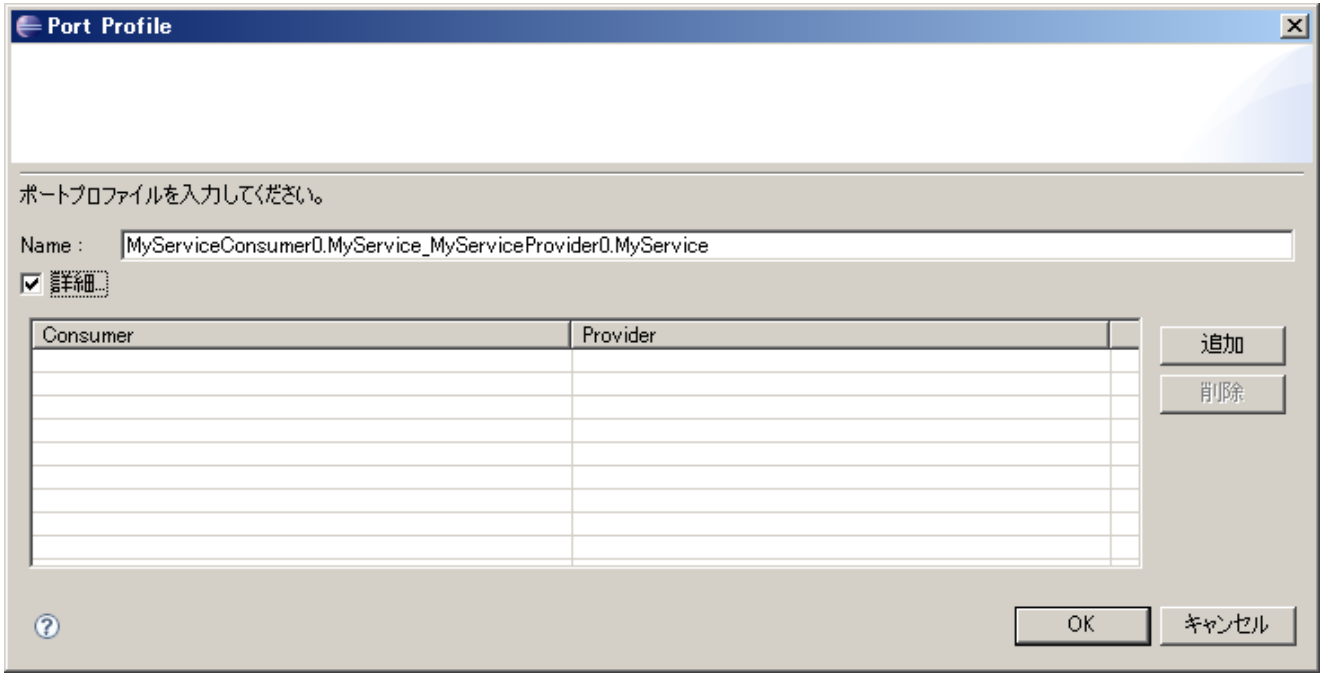
※OutPort側のバッファ，InPort側のバッファそれぞれに設定可能
※timeoutとして「0.0」を設定した場合は，タイムアウトしない

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

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

接続プロファイル(ServicePort)について

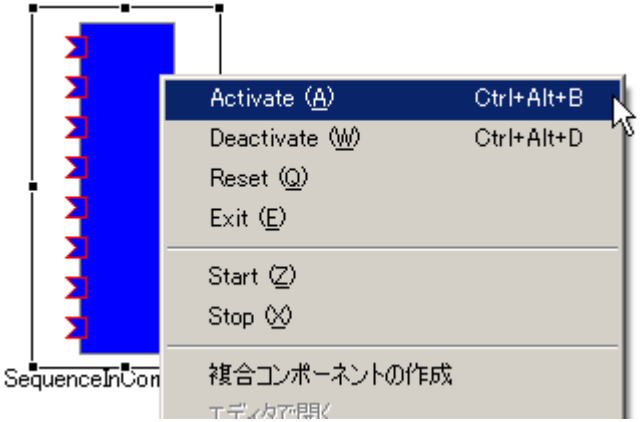
項目	設定内容
Name	接続の名称
インターフェース情報	接続するインターフェースを設定. 接続対象のServicePortに複数のServiceInterfaceが定義されていた場合，どのインターフェースを実際に接続するかを指定



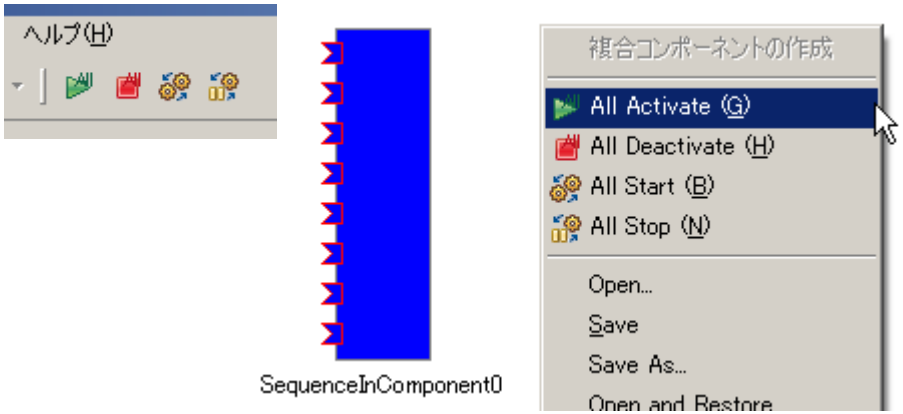
RTコンポーネントの動作

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



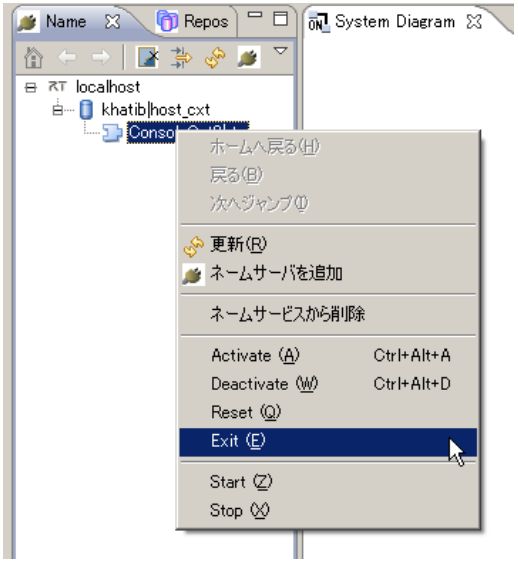
■全コンポーネントの動作を一括変更

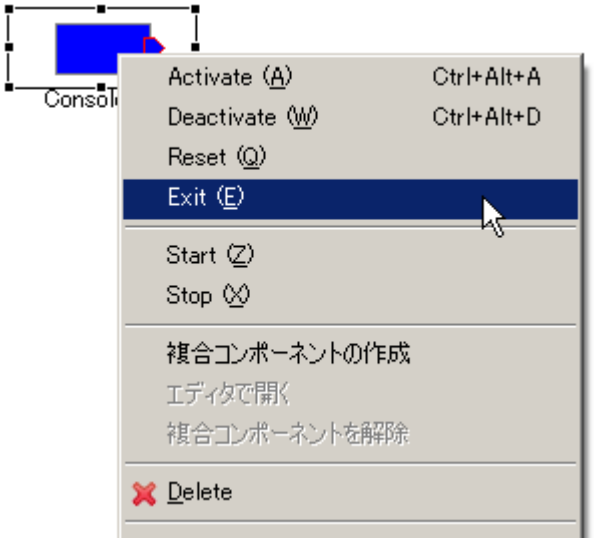


※ポップアップメニュー中でのキーバインドを追加
※単独RTCのActivate/Deactivateについては，グローバルはショートカットキー定義を追加

サンプルの切り替え

- ConsoleIn コンポーネント/ConsoleOut コンポーネントの終了
  - 入力コンソールを終了
  - ネームサービスビュー内で対象RTCを選択，右クリックし「Exit」
  - システムエディタ上で対象RTCを選択，右クリックし「Exit」



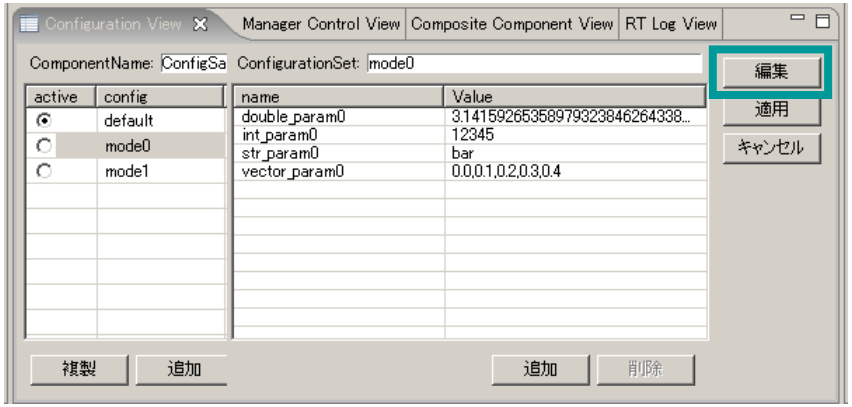


- RTC daemonの起動
  - [スタート]メニューから起動

[プログラム] →[OpenRTM-aist]→[C++]→[tools]→[Start RTC daemon]

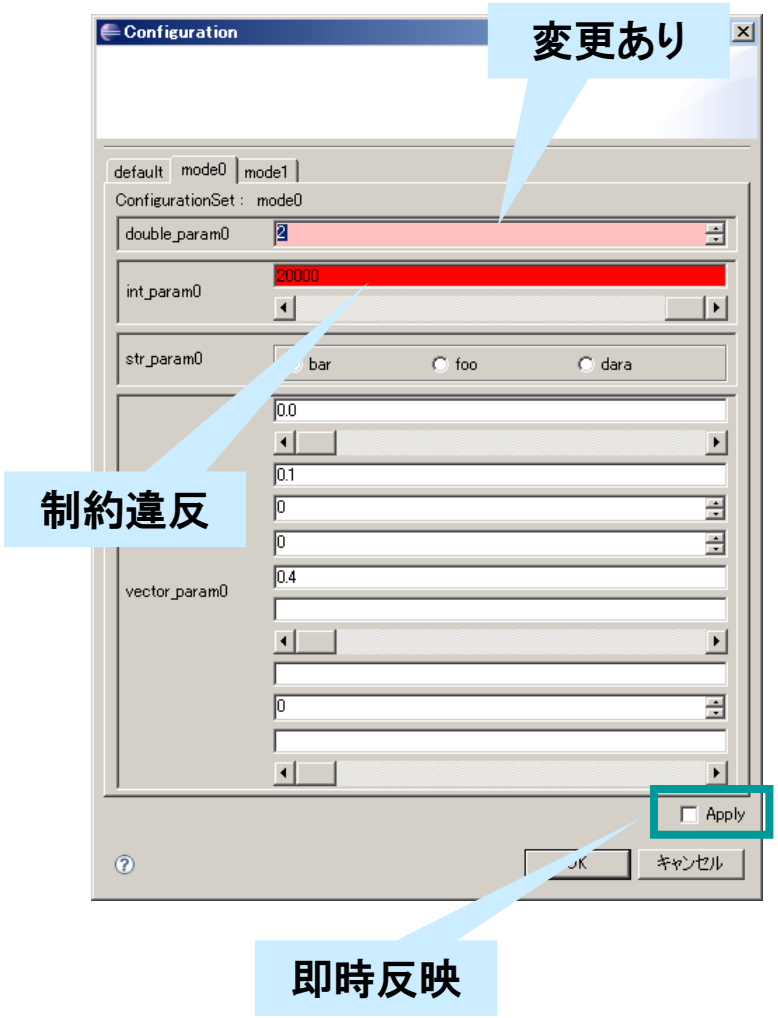
コンフィギュレーションビュー

- 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，
スライドバーを操作する，
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)

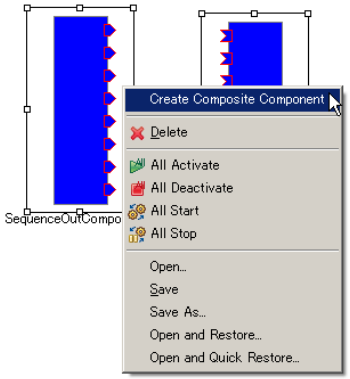
RTCの利用者が設定するのではなく，RTC開発者，RTC管理者が設定することを想定．

RTCBuilderを使用することで設定可能

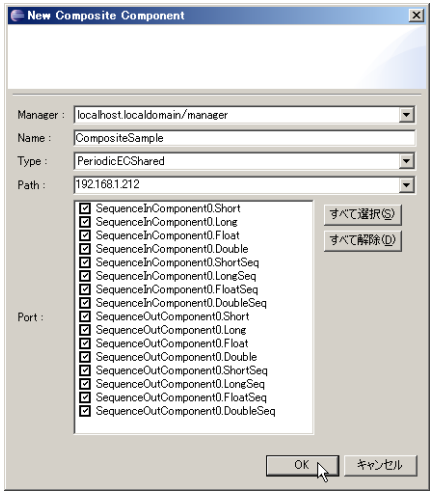
複合コンポーネント

- 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

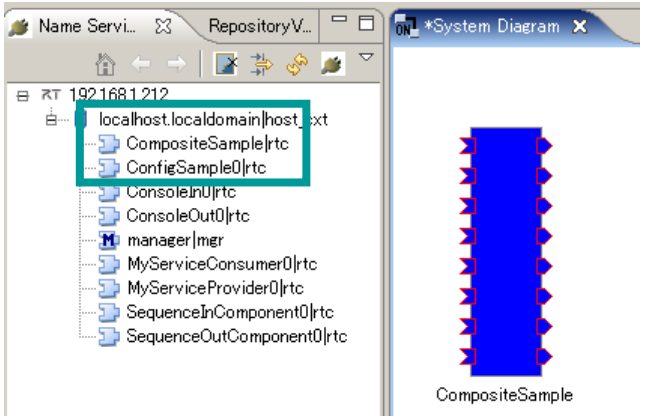
①複数RTCを選択している状態で右クリック



②複合コンポーネントのプロパティを設定



③複合コンポーネントを生成



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

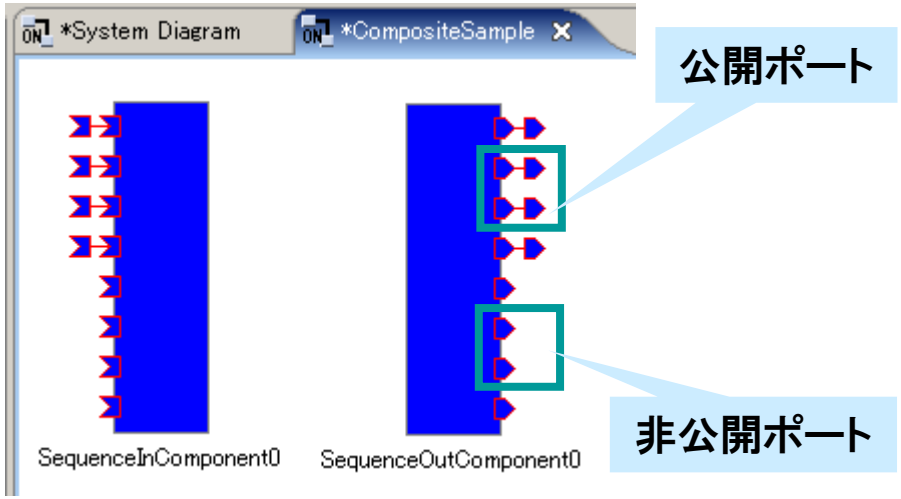
※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

複合コンポーネント

- 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



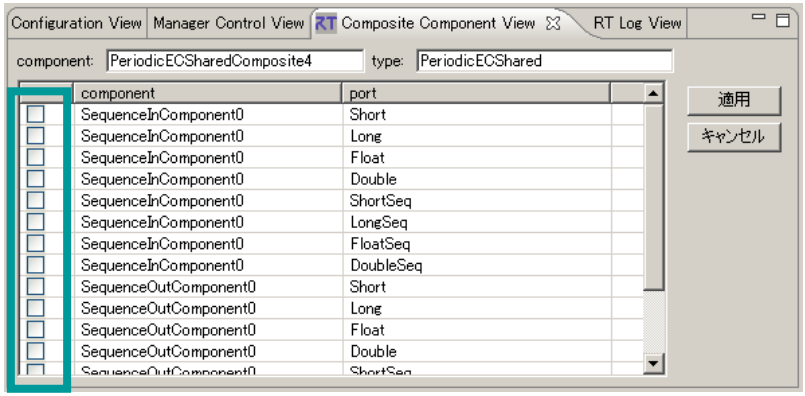
※エディタ内に別RTCをDnDすることで，
子コンポーネントの追加が可能
→追加したRTCのポートは
全て非公開に設定
※エディタ内のRTCを削除することで，
子コンポーネントの削除が可能
→削除されたRTCは，親エディタに
表示

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

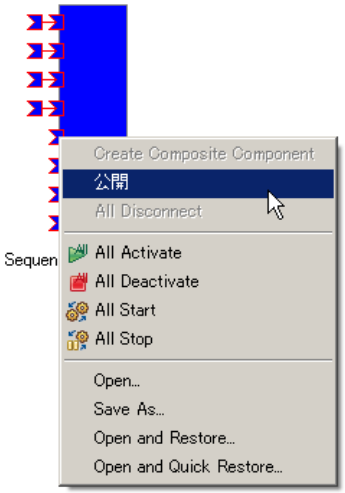
ポート公開情報



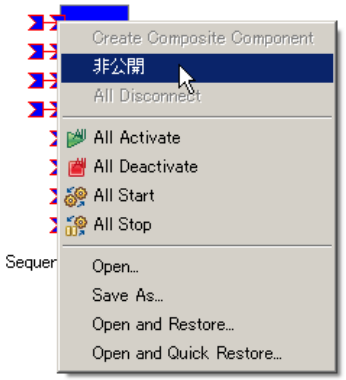
※ポート公開情報を変更し，「適用」をクリック

- 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」



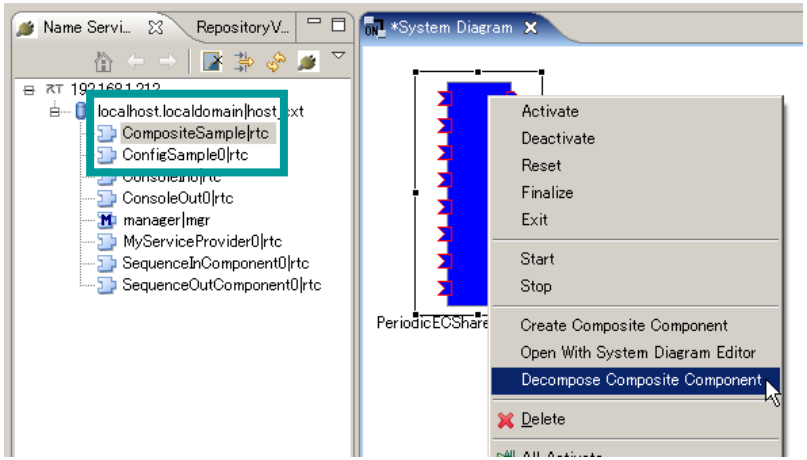
外部コンポーネントと接続されているポートを「非公開」に設定することはできません

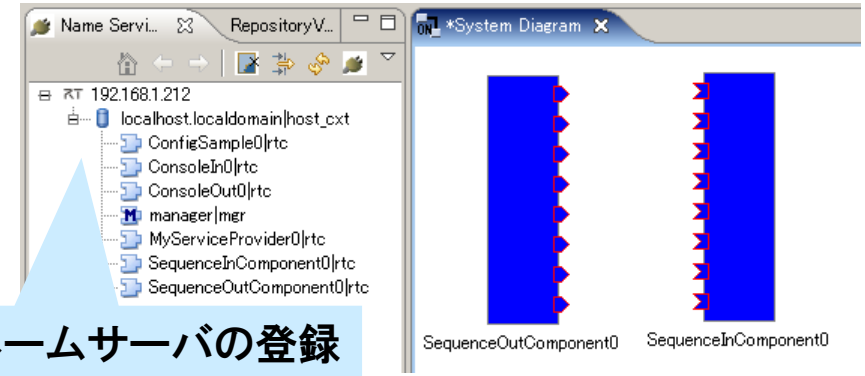
複合コンポーネント

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





ネームサーバの登録も解除される

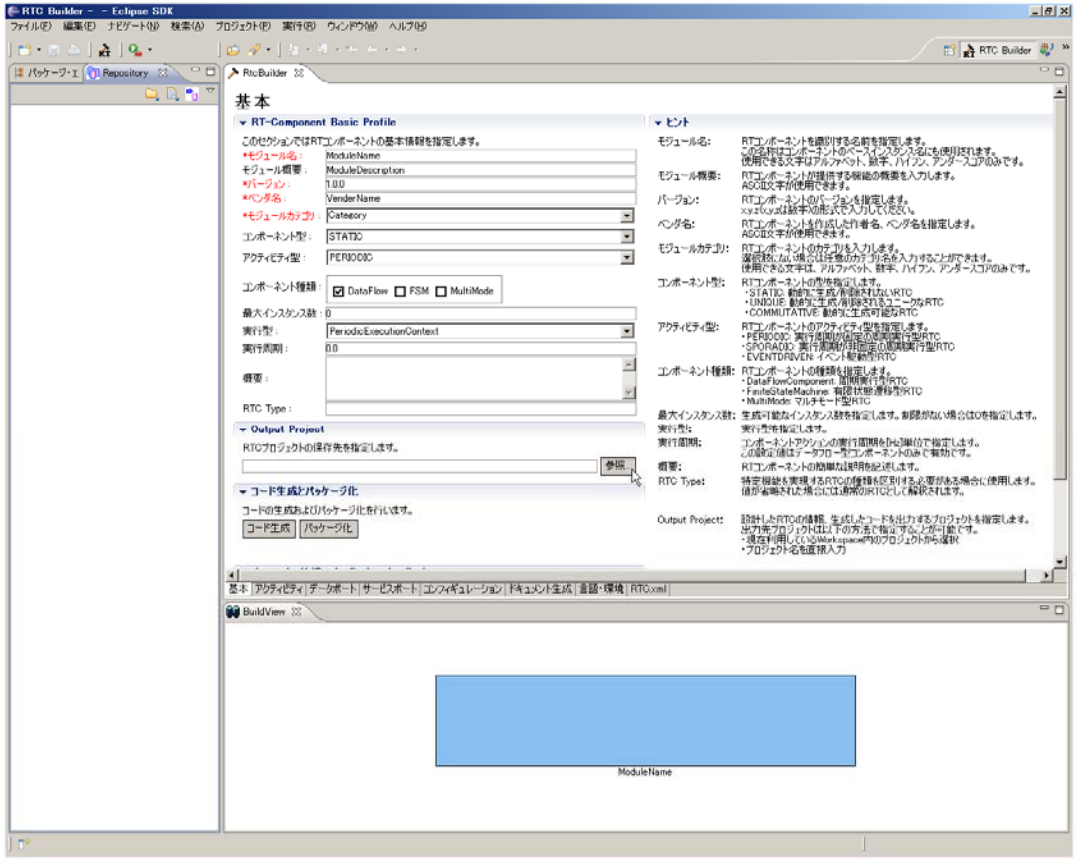
※エディタ上で，（Deleteキーなどで）単純に削除した場合は，エディタから表示が消えるのみ
複合コンポーネントは解除されない

RTCBuilderについて

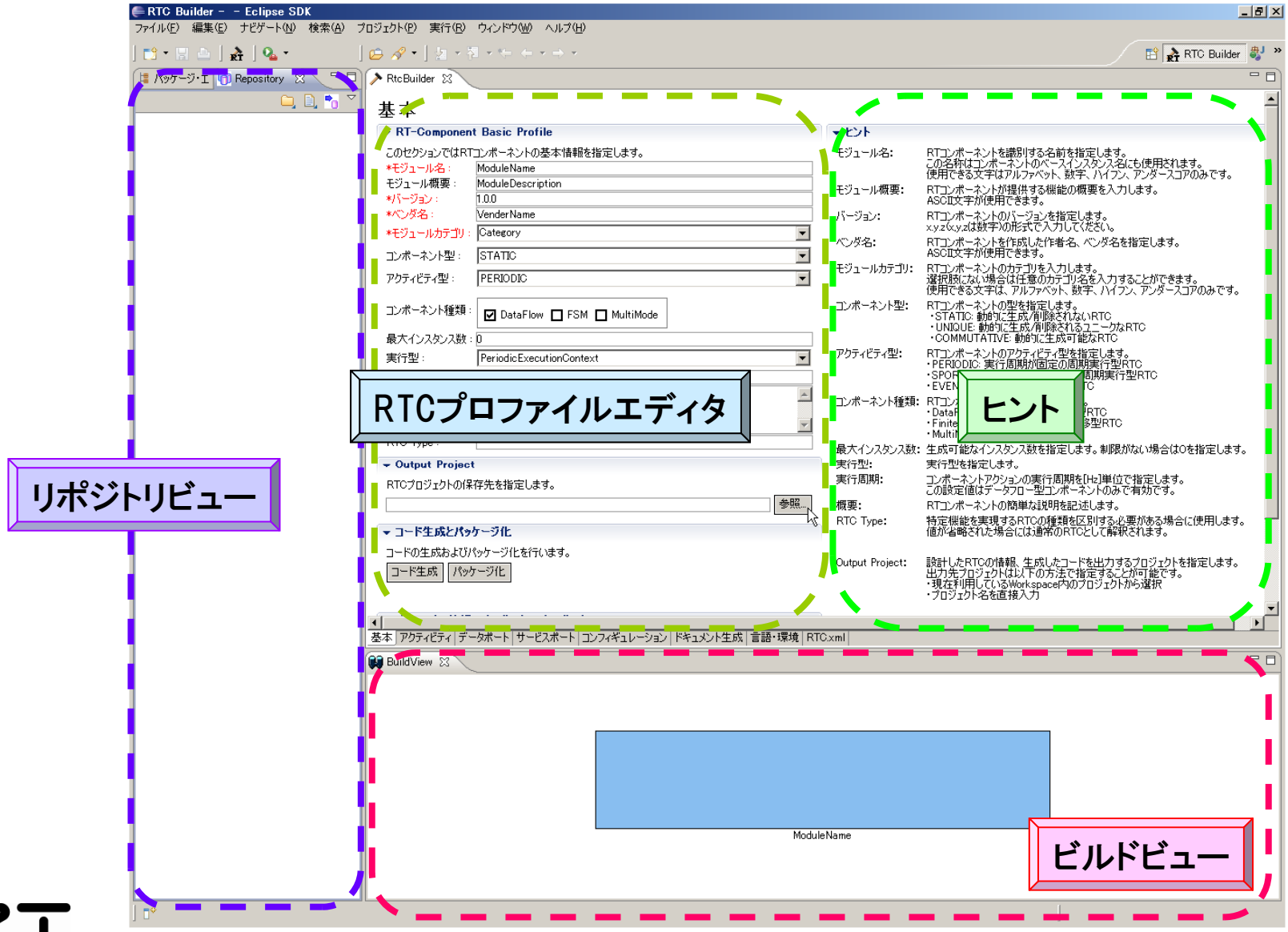
RTCBuilder概要

- 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



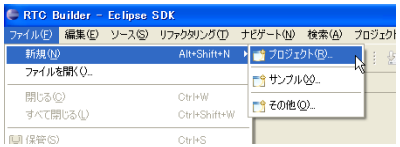
画面構成


RTC Builder - - Eclipse SDK
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
RTC Builder
パッケージ・エ
Repository
RtcBuilder
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 : ModuleName
モジュール概要 : ModuleDescription
*バージョン : 1.0.0
*ベンダ名 : VenderName
*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 : 0
実行型 : PeriodicExecutionContext
Output Project
RTCプロジェクトの保存先を指定します。
参照...
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
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。
x.y.z(x,y,zは数字)の形式で入力してください。
ベンダ名: RTコンポーネントを作成した作者名、ベンダ名を指定します。
ASCII文字が使用できます。
モジュールカテゴリ: RTコンポーネントのカテゴリを入力します。
選択肢にない場合は任意のカテゴリ名を入力することができます。
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。
コンポーネント型: RTコンポーネントの型を指定します。
・STATIC: 動的に生成/削除されないRTC
・UNIQUE: 動的に生成/削除されるユニークなRTC
・COMMUTATIVE: 動的に生成可能なRTC
アクティビティ型: RTコンポーネントのアクティビティ型を指定します。
・PERIODIC: 実行周期が固定の周期実行型RTC
コンポーネント種類: RTコン
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ます。制限がない場合は0を指定します。
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を[Hz]単位で指定します。
この設定値はデータフロー型コンポーネントのみで有効です。
概要: RTコンポーネントの簡単な説明を記述します。
RTC Type: 特定機能を実現するRTCの種類を区別する必要がある場合に使用します。
値が省略された場合には通常のRTCとして解釈されます。
Output Project: 設計したRTCの情報、生成したコードを出力するプロジェクトを指定します。
出力先プロジェクトは以下の方法で指定することが可能です。
・現在利用しているWorkspace内のプロジェクトから選択
・プロジェクト名を直接入力
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
BuildView
ModuleName
RTCプロファイルエディタ
ヒント
リポジトリビュー
ビルドビュー

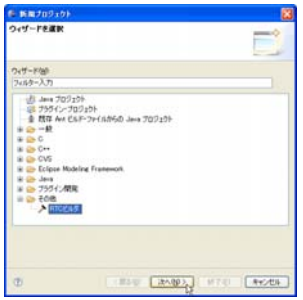
プロジェクト作成/エディタ起動

※パースペクティブを「RtcBuilder」に切り替え

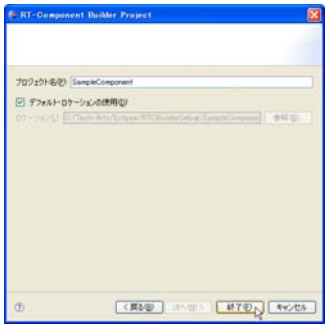
①メニューから「ファイル」ー「新規」ー「プロジェクト」



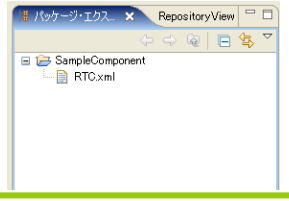
②「その他」-「RtcBuilder」を選択し，「次へ」



③「プロジェクト名」欄に入力し，「終了」



④指定した名称のプロジェクトを生成



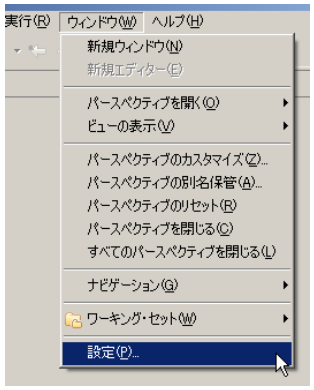
※任意の場所にプロジェクトを作成したい場合
③にて「デフォルト・ロケーションの使用」チェックボックスを外す
「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名：Flip

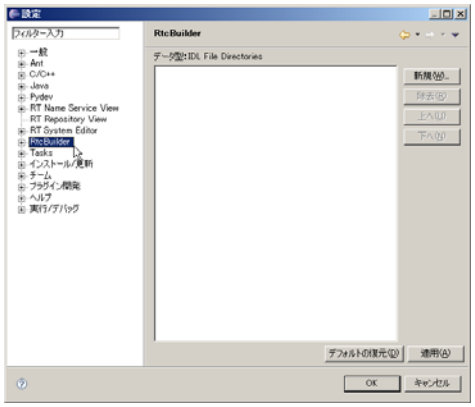
各種設定・起動

- 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

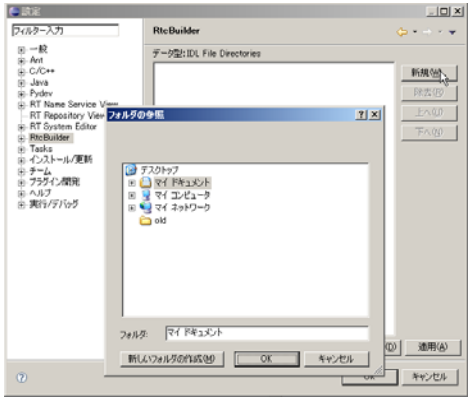
①メニューから
「ウインドウ」ー「設定」



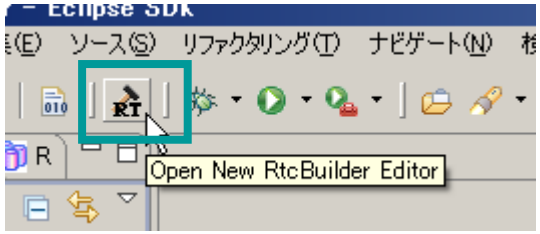
②「RtcBuilder」を選択

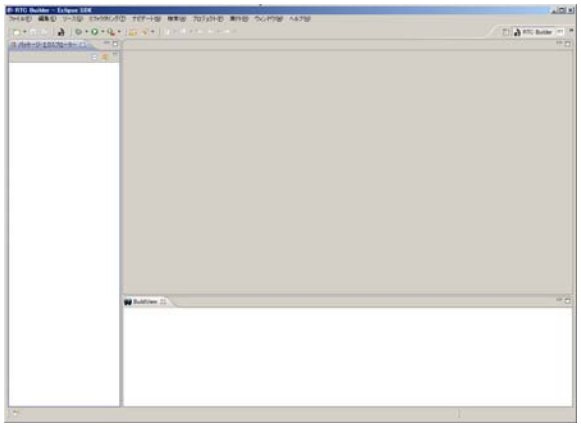


③「新規」ボタンにて表示される
ディレクトリ選択ダイアログ
にて場所を指定



- RtcBuilderの起動
  画面上部のアイコンをクリック





RTCプロファイルエディタ

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

基本プロファイル

- RTコンポーネントの名称など，基本的な情報を設定

モジュール名: Flip
モジュール概要: 任意(Flip image component)
バージョン: 1.0.0
ベンダ名: 任意(AIST)
モジュールカテゴリ: 任意(Category)
コンポーネント型: STATIC
アクティビティ型: PERIODIC
コンポーネントの種類: DataFlow
最大インスタンス数: 1
実行型: PeriodicExecutionContext
実行周期: 0.0
Output Project: Flip

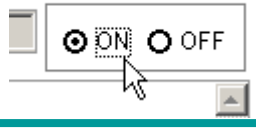
※エディタ内の項目名が赤字の要素は必須入力項目
※画面右側は各入力項目に関する説明

アクティビティ・プロファイル

- 生成対象RTCで実装予定のアクティビティを設定

①設定対象のアクティビティを選択

②使用/未使用を設定



以下をチェック：
on_activated
on_deactivated
on_execute

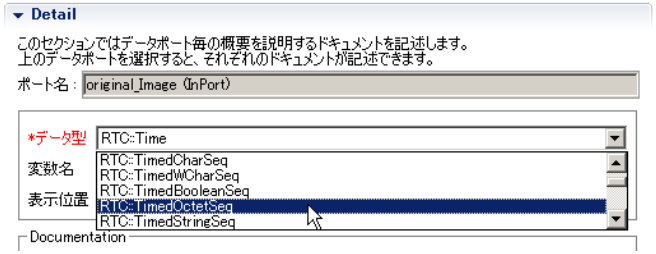
※現在選択中のアクティビティは，一覧画面にて赤字で表示
※使用(ON)が選択されているアクティビティは，一覧画面にて背景を水色で表示
※各アクティビティには，「動作概要」「事前条件」「事後条件」を記述可能
→記述した各種コメントは，生成コード内にDoxygen形式で追加される

データポート・プロファイル

生成対象RTCに付加するDataPortの情報を設定

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

②設定する型情報を一覧から選択



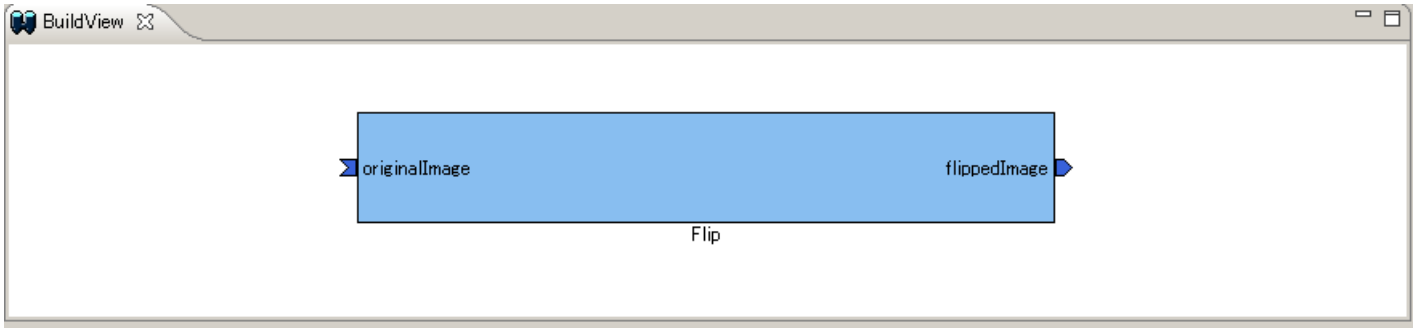
※データ型は，型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

※OpenRTM-aistにて事前定義されている型については，デフォルトで使用可能
→[OpenRTM_Root]/1.0/rtm/idl 以下に存在するIDLファイルで定義された型

※各ポートに対する説明記述を設定可能
→記述した各種コメントは，生成コード内にDoxygen形式で追加される

データポート・プロファイル

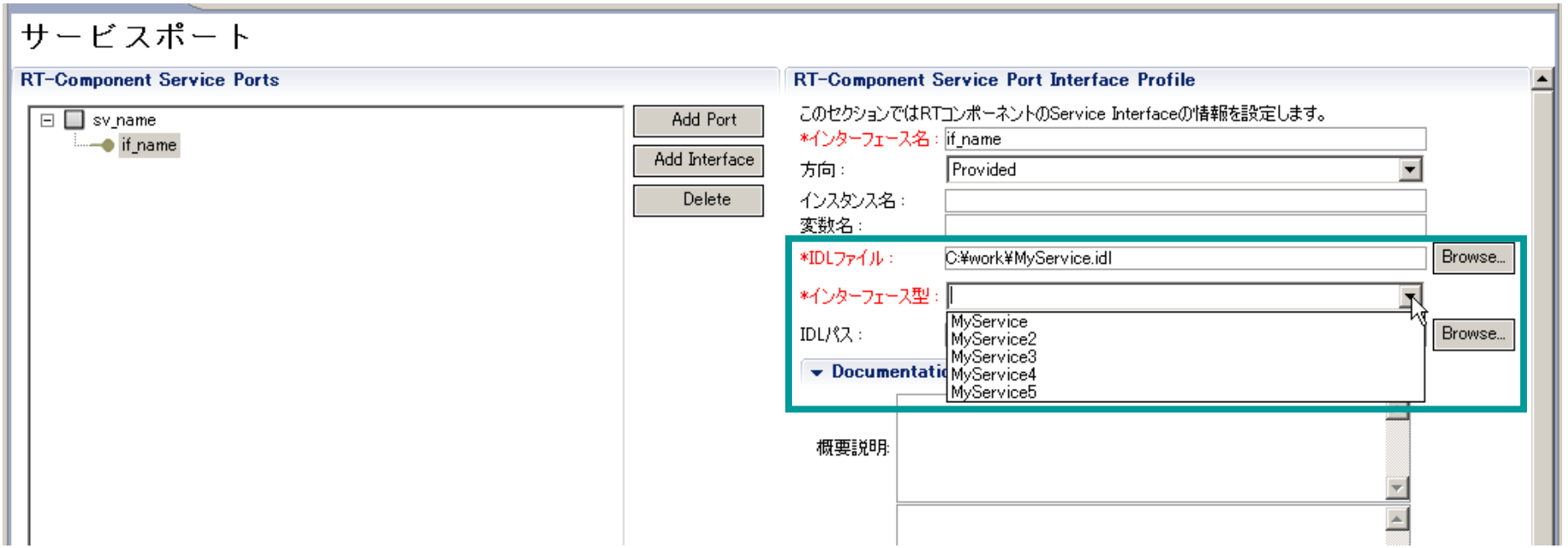
※Portの設定内容に応じて，下部のBuildViewの表示が変化



- InPort
  - ポート名: originalImage
  - データ型: RTC::CameraImage
  - 変数名: originalImage
  - 表示位置: left
- OutPort
  - ポート名: flippedImage
  - データ型: RTC::CameraImage
  - 変数名: flippedImage
  - 表示位置: right

サービスポート・プロファイル

- 生成対象RTCに付加するServicePortの情報を設定

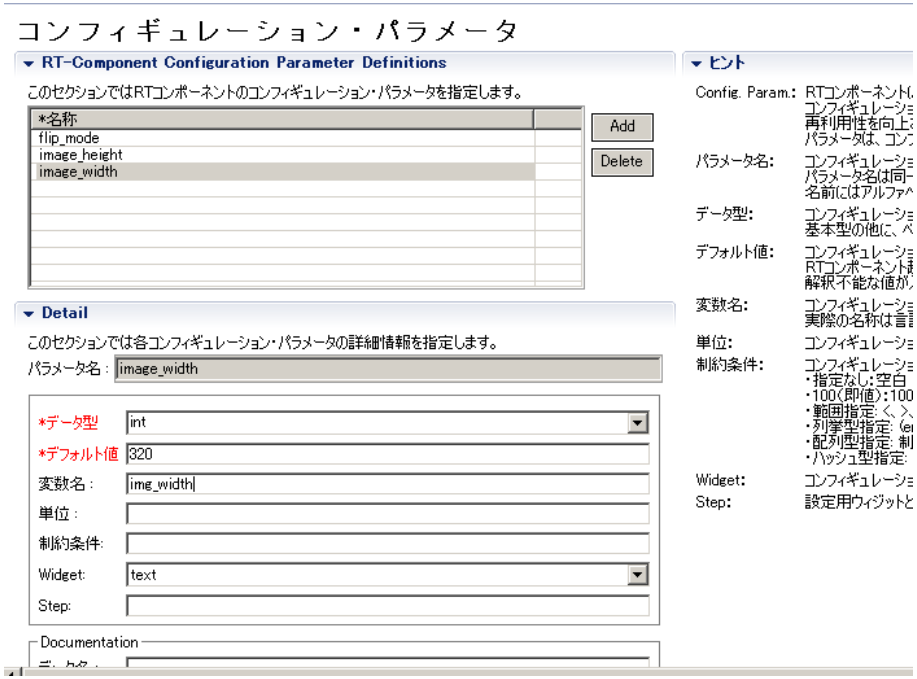


- サービスインターフェースの指定
  - IDLファイルを指定すると，定義されたインターフェース情報を表示

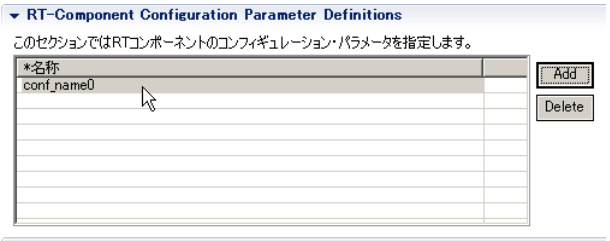
今回のサンプルでは未使用

コンフィギュレーション・プロファイル

- 生成対象RTCで使用する設定情報を設定



①「Add」ボタンをクリックし，追加後，
直接入力で名称設定



②詳細画面にて，型情報，変数名などを設定

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

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

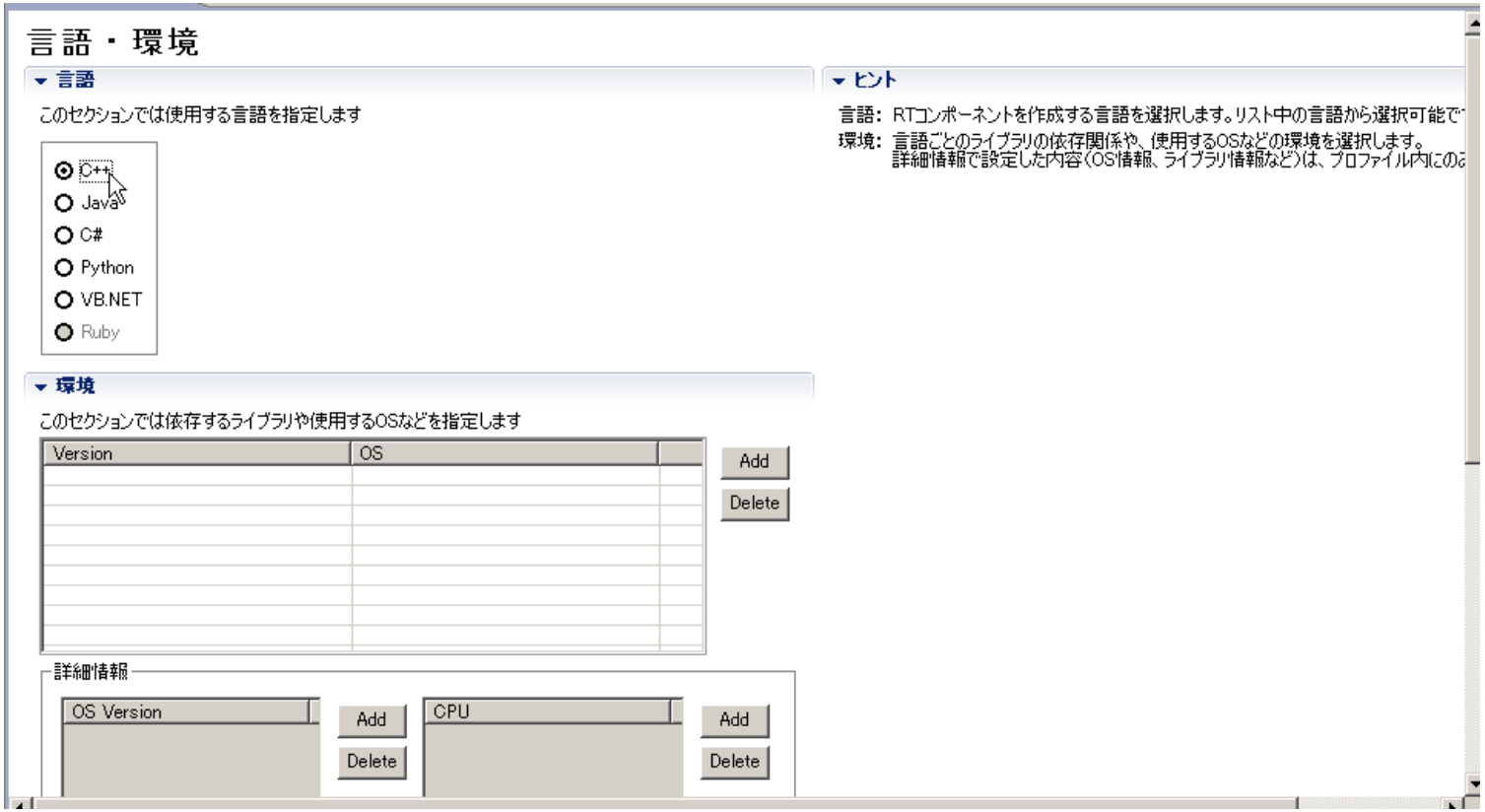
制約条件，Widgetの設定方法

- 制約条件について
  - データポートとコンフィギュレーションに設定可能
  - チェックはあくまでもコンポーネント開発者側の責務
    - ミドルウェア側で検証を行っているわけではない
- 制約の記述書式
  - 指定なし：空白
  - 即値：値そのもの
    - 例) 100
  - 範囲：＜，＞，＜＝，＞＝
    - 例) 0<=x<=100
  - 列挙型：（値1，値2，・・・）
    - 例) (val0, val1, val2)
  - 配列型：値1，値2，・・・
    - 例) val0, val1, val2
  - ハッシュ型：{ key0：値0， key1：値1，・・・ }
    - 例) { key0:val0， key1:val1}
- Widget
  - text（テキストボックス）
    - デフォルト
  - slider（スライダ）
    - 数値型に対して範囲指定の場合
    - 刻み幅をstepにて指定可能
  - spin(スピナ)
    - 数値型に対して範囲指定の場合
    - 刻み幅をstepにて指定可能
  - radio(ラジオボタン)
    - 制約が列挙型の場合に指定可能

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

言語・環境・プロファイル

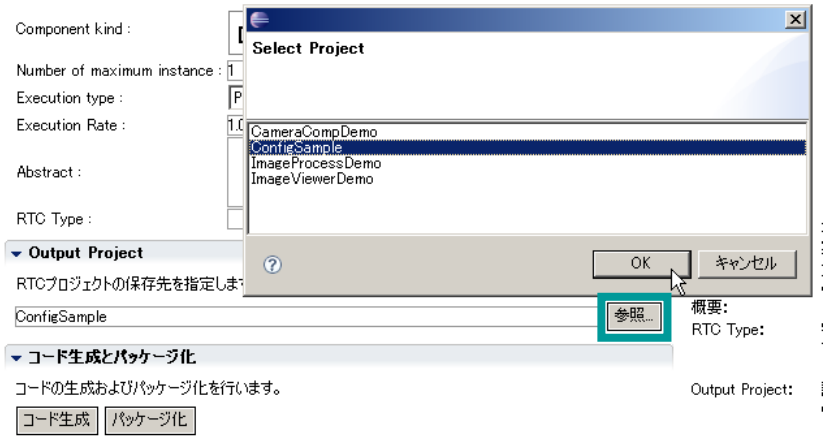
- 生成対象RTCを実装する言語，動作環境に関する情報を設定



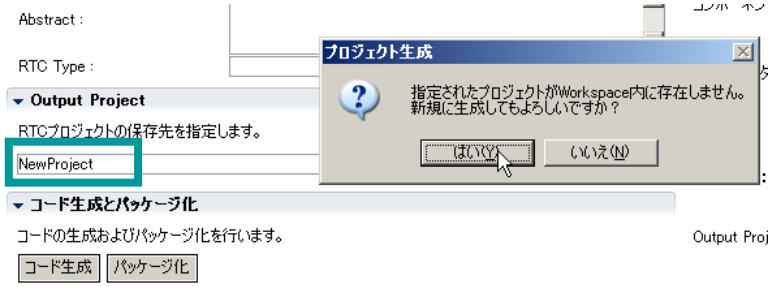
「C++」を選択

コード生成

出力先プロジェクト選択

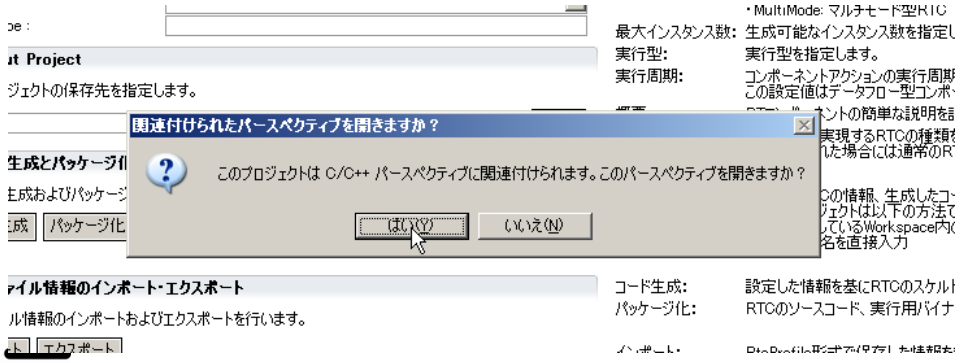


※ワークスペース内のプロジェクトから選択



※プロジェクト名を直接入力
→該当プロジェクトがワークスペース内に存在しない場合，新規作成することも可能

コード生成実行後，パースペクティブを自動切替



C++版RTC → CDT

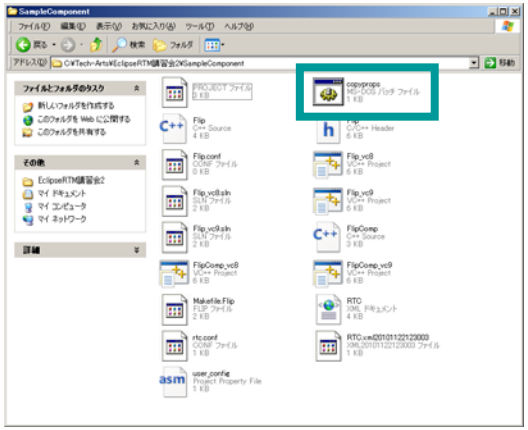
Java版RTC → JDT

（デフォルトインストール済み）

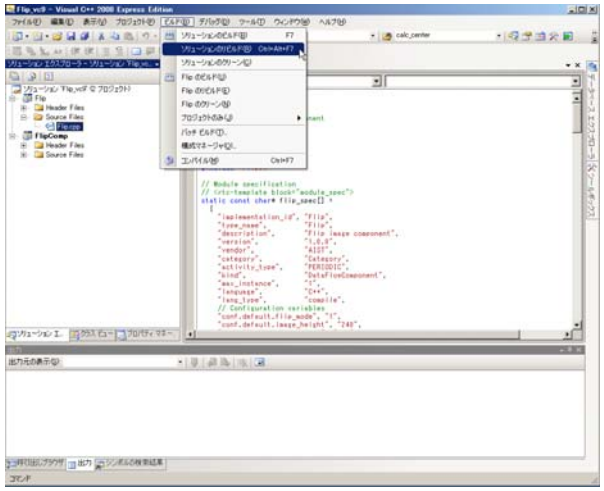
Python版 → PyDev

コンパイル・実行

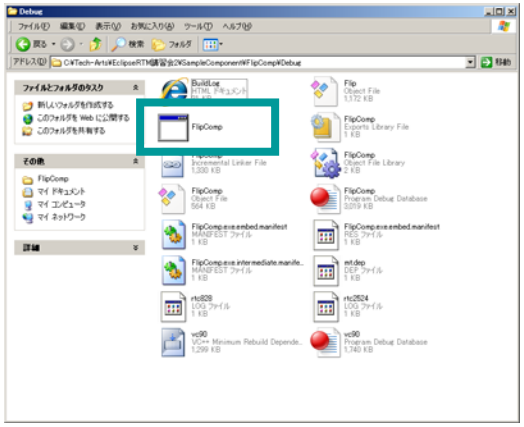
①コード生成先ディレクトリ内の
「copyprops.bat」をダブルクリックして，
設定ファイルをコピー

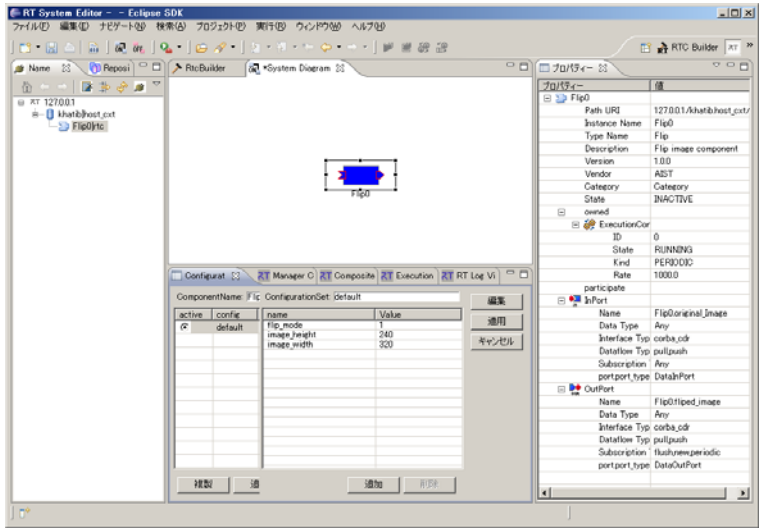


②VisualStudioを用いたビルド



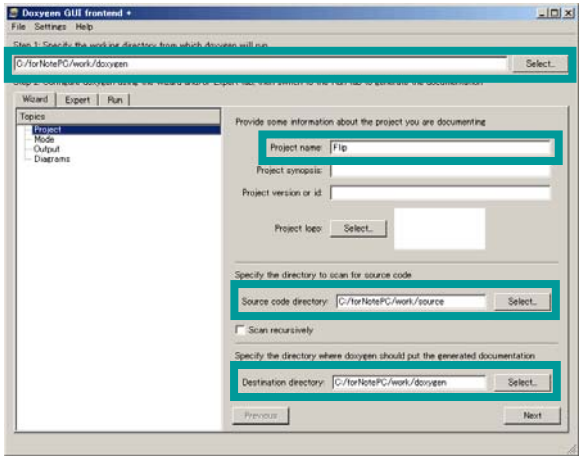
③FlipComp¥¥Debug内のFlipComp.exeを起動



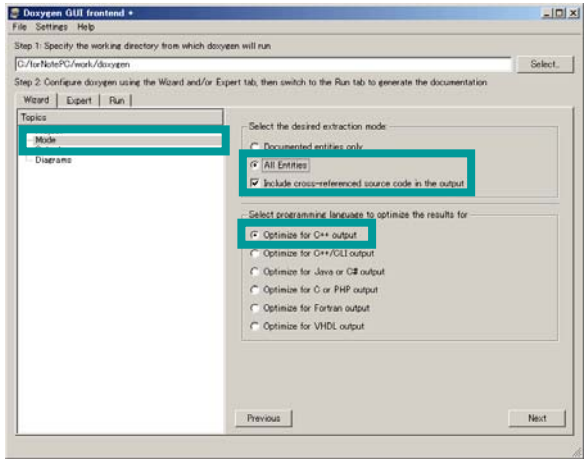


ドキュメント作成

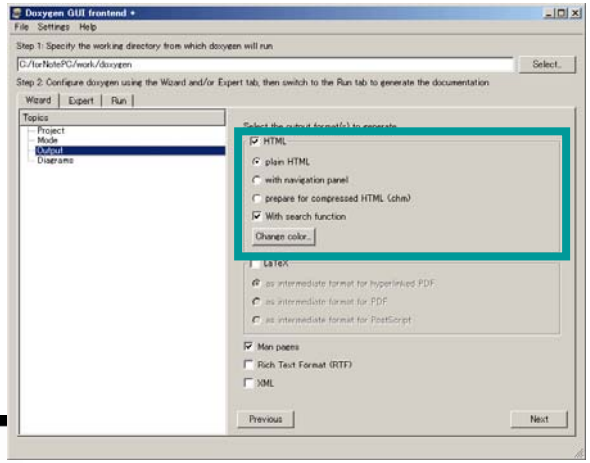
①Doxygen用GUIツールを起動
作業用ディレクトリ,ソース格納場所,
生成ファイル出力先,プロジェクト名を指定



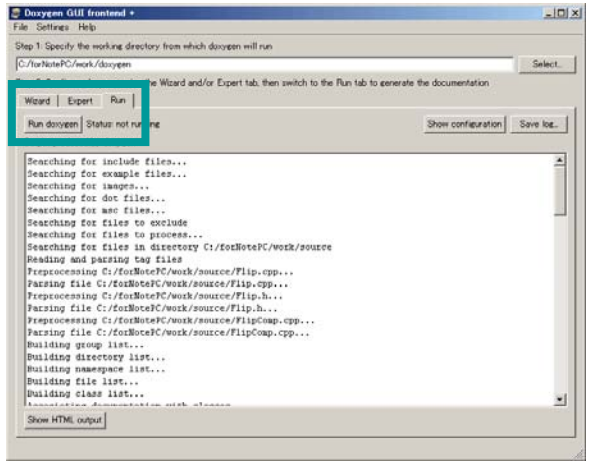
②「Mode」セクションにて,
出力内容,使用言語を指定



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



ドキュメント作成

生成されたドキュメントの例

Flip

Main Page | Classes | Files | Search

Class List | Class Index | Class Members

Public Member Functions | Protected Attributes

Flip Class Reference

Flip image component. More...

`#include <Flip.h>`

List of all members.

Public Member Functions

	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)

Protected Attributes

int	**m_flipMode**
CameraImage	**m_originalImage**
InPort< CameraImage >	**m_originalImageIn**
CameraImage	**m_flippedImage**
OutPort< CameraImage >	**m_flippedImageOut**

Detailed Description

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使用してflipModeという名前のパラメータで指定
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。
•InPort:キャプチャされた画像データ(CameraImage)

Member Function Documentation

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の画面サイズの初期化

Definition at line **129** of file **Flip.cpp**.

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

Definition at line **139** of file **Flip.cpp**.

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

Definition at line **152** of file **Flip.cpp**.

RTC::ReturnCode_t Flip::onInitialize ( ) [virtual]

コンポーネント自身の各種初期化処理

Definition at line **76** of file **Flip.cpp**.

Member Data Documentation

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
- Range: -1,0,1
- Constraint: 0: 上下反転したい場合
  1: •左右反転したい場合
  -1: 上下左右反転したい場合

Definition at line **297** of file **Flip.h**.

Flip

Main Page | Classes | Files

File List | File Members

C:/forNotePC/work/source/Flip.cpp

Go to the documentation of this file.

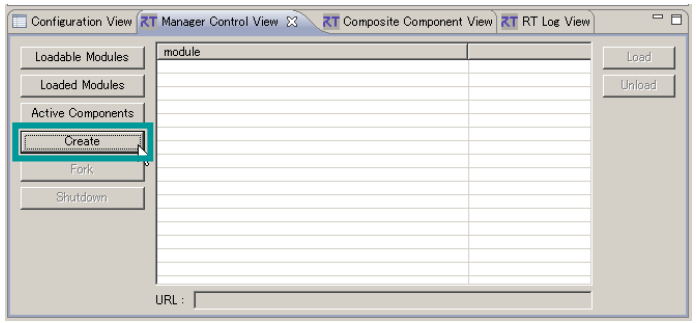
```
00001 // -*- C++ -*-
00024 #include "Flip.h"
00025
00026 // Module specification
00027 // <rtc-template block="module_spec">
00028 static const char* flip_spec[] =
00029   {
00030     "implementation_id", "Flip",
00031     "type_name",         "Flip",
00032     "description",       "Flip image component",
00033     "version",           "1.0.0",
00034     "vendor",            "AIST",
00035     "category",          "Category",
00036     "activity_type",     "PERIODIC",
00037     "kind",              "DataFlowComponent",
00038     "max_instance",      "1",
00039     "language",          "C++",
00040     "lang_type",         "compile",
00041     // Configuration variables
00042     "conf.default.flipMode", "0",
00043     // Widget
00044     "conf.__widget__.flipMode", "radio",
00045     // Constraints
00046     "conf.__constraints__.flipMode", "(-1,0,1)",
00047     ""
00048   };
00049 // </rtc-template>
00050
00055 Flip::Flip(RTC::Manager* manager)
00056     // <rtc-template block="initializer">
00057   : RTC::DataFlowComponentBase(manager),
00058     m_originalImageIn("originalImage", m_originalImage),
00059     m_flippedImageOut("flippedImage", m_flippedImage)
00060
00061     // </rtc-template>
00062 {
00063 }
00064
00068 Flip::~Flip()
00069 {
00070 }
```

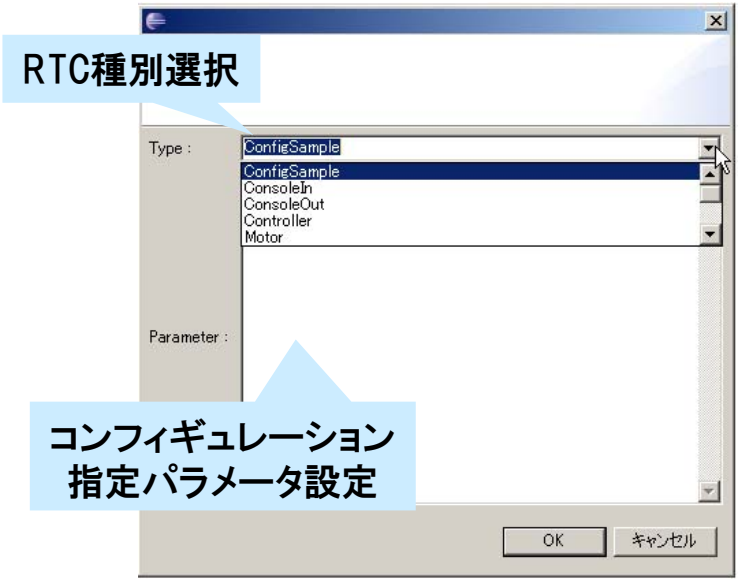
RTSystemEditorの補足

株式会社テクノロジックアート
TECHNOLOGIC ARTS INCORPORATED

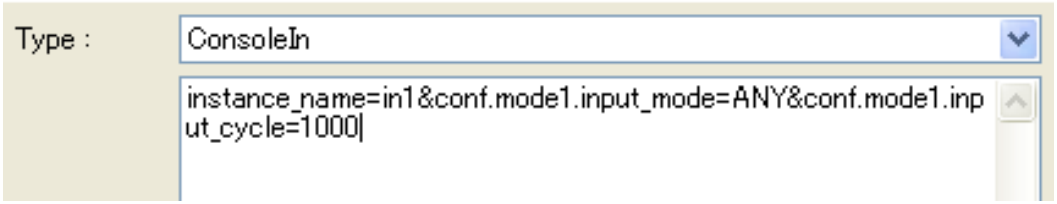
マネージャビュー

- RTコンポーネントの新規インスタンスの生成



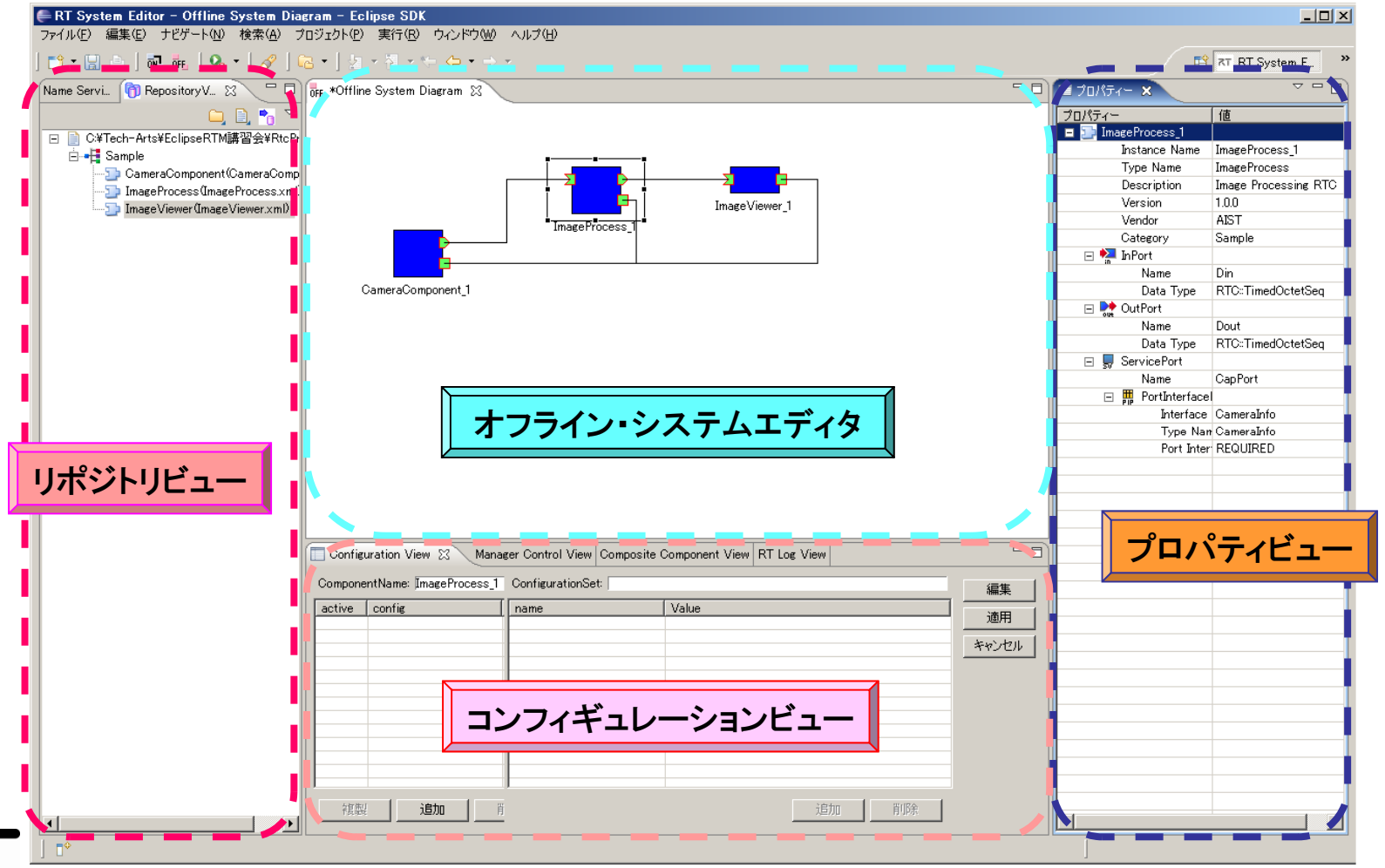


- コンフィギュレーション指定パラメータ
  - conf.[ConfigSet名].[Configパラメータ名]=[設定値]の形式にてConfigurationSetの値も設定可能



オフラインエディタ

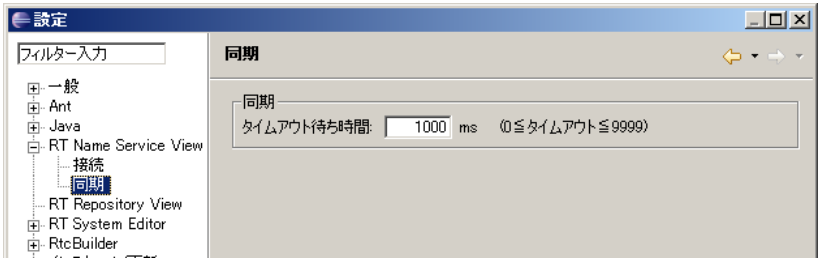
- RTコンポーネントの仕様を用いてRTシステムを構築
  - 実際のRTコンポーネントが動作している必要はない



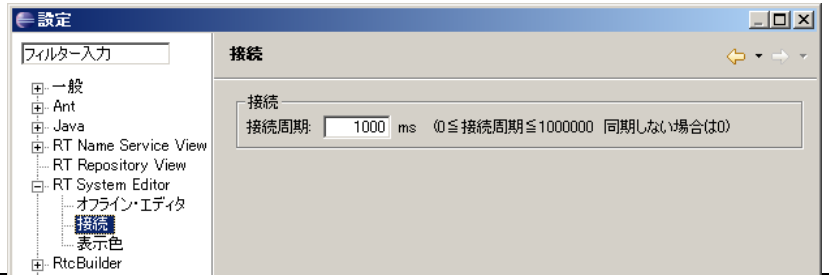
設定画面

- 「RT Name Service View」ー「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」ー「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」ー「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

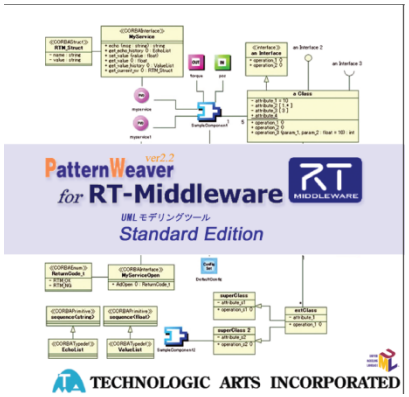
その他のツールのご紹介

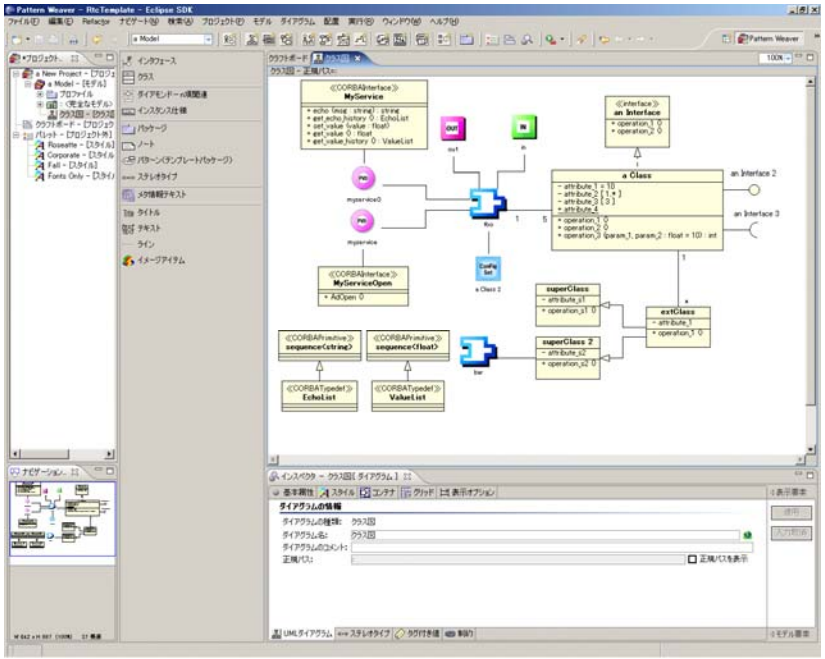
株式会社テクノロジックアート
TECHNOLOGIC ARTS INCORPORATED

RT MIDDLEWARE

UMLを利用した設計情報から，RTCの雛形コードを生成

- OpenRTM-aist-1.0.0-RELEASEに対応
  - C++, Java, Python
- UMLを用いたコンポーネント設計/システム設計支援
- システム設計情報を基にした各種コードの自動生成
  - RTコンポーネント(C++, Java, Python)雛形コード
  - CORBA IDLコード
  - 関連クラス(C++, Java, Python)雛形コード





- RTC，RTシステム開発支援ツールのご提供
  - RTシステムローダー
  - プラグアンドプレイ設定ツール
  - 状態遷移設定ツール

※PatternWeaverサイト　http://pw.tech-arts.co.jp/

お問い合わせ先　pw@tech-arts.co.jp

株式会社テクノロジックアート
TECHNOLOGIC ARTS INCORPORATED

システム構築ツールセット

- RTコンポーネント，RTシステムを制御するためのPythonライブラリ群
  - 各種操作の自動化，プログラムからの利用を可能に

- rtctree
  - 簡単なAPIでRTコンポーネントを管理するためのライブラリ
  - CORBAのAPIを知らなくても，他プログラムからRTCを管理可能
  - RTCのactivate/deactivate，ポート間の接続を行うこと等が可能
- rtcshell
  - ネームサーバに登録されているRTコンポーネントをシェルから管理するツール
  - RTCのactivate/deactivate，ポート間の接続を行うこと等が可能
  - リソースの少ないシステム，GUIが利用できない環境でも利用可能
- rtsprofile
  - RTシステム仕様(RTSProfile)のインタフェースライブラリ
  - システムの復元や管理が可能
  - XML，YAMLを利用可能

システム構築ツールセット

- rtsshell
  - rtresurrect
    - RTSProfileに保存されたRTシステムを復元するためのツール
    - 全ての接続，コンフィギュレーション設定を復元し，アクティブなコンフィギュレーションを設定可能
  - rtteardown
    - RTSProfileに保存されたRTシステムをシャットダウンするツール
    - 全ての接続を切断
  - rtcryo
    - 既存のRTシステムをRTSProfileのファイルとして保存するツール
    - 全てのコンポーネント，コンポーネント間の接続，構成設定をファイルに保存
  - rtstart/rtstop
    - コマンドラインでRTシステムを起動/停止するツール
    - RTSProfileに記載された実行順，実行条件に従ってRTコンポーネントを制御

RTミドルウエア講習会

株式会社テクノロジックアート
TECHNOLOGIC ARTS INCORPORATED