RTミドルウエア(OpenRTM-aist-1.1.0) 講習会

日時:2011年5月26日(木)　10:00～17:00
場所:岡山コンベンションセンター 4F 405会議室

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

RT MIDDLEWARE

RTミドルウェア講習会

10:00-10:50	**第1部(その1):OpenRTM−aist-1.1.0の新機能と今後の展望について**
	担当:神徳 徹雄(産業技術総合研究所)
	概要:OpenRTM-aist-1.1.0の新機能および、今後の展望について。
11:00-11:50	**第1部(その2):インターネットを利用したロボットサービスとRsiの取り組み(最新動向)**
	担当:成田 雅彦(産業技術大学院大学)
	概要:RSi(Robot Service Initiative)およびRSNP(Robot Service Network Protocol)について解説します。
11:50-12:00	**質疑応答・意見交換**
13:00-13:30	**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(産業技術総合研究所)
	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 していただきます。
13:30-14:00	**第3部:OpenRTM-aist コマンドラインツール rtshell 利用法**
	担当:Geoffrey Biggs(産業技術総合研究所)
	概要:RTコンポーネントをコマンドラインから操作するツール rtshellの使い方について解説します。
14:10-15:00	**第4部:OpenRTM-aist開発支援ツールの紹介とその利用方法**
	担当:坂本 武志(テクノロジックアート)
	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RTSystemEditorの使い方について解説します。
15:15-17:00	**第5部:コンポーネント開発実習**
	担当:片見 剛人(富士ソフト株式会社)
	概要: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RTSystemEditorでのRTシステム作成を行います。

第4部　OpenRTM-aist開発支援ツールの紹介とその利用法

株式会社　テクノロジックアート
ロボット事業推進グループ
坂本 武志

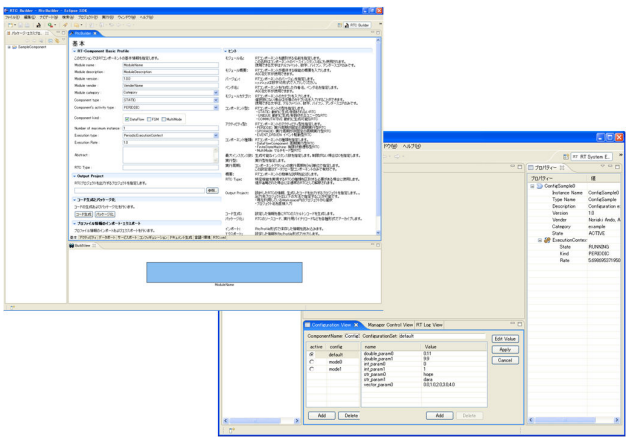
OpenRTM-aistの開発支援ツールについて

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

OpenRT Platform

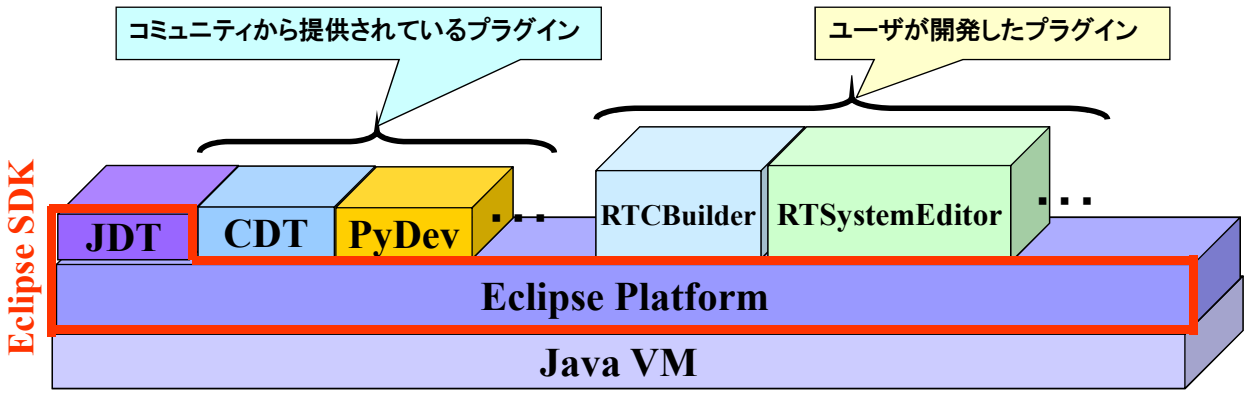
- 次世代ロボット知能ソフトウェアプラットフォーム
  - http://www.openrtp.jp/wiki/
  - システム設計，シミュレーション，動作生成，シナリオ生成などをサポート
- OpenRT Platformツール群
  - コンポーネント開発，システム開発における各開発フェーズの作業支援
  - 開発プラットフォームにEclipseを採用
- 構成
  - RTCビルダ
  - RTCデバッガ
  - RTシステムエディタ
  - ロボット設計支援ツール
  - シミュレータ
  - 動作設計ツール
  - シナリオ作成ツール

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



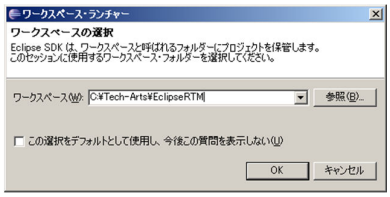
RTCBuilder,RTSystemEditorのインストール

- 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

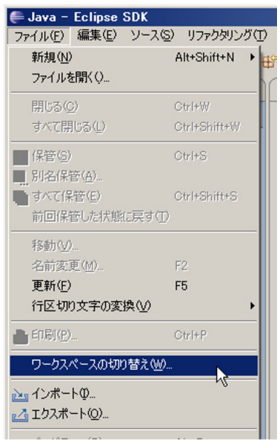
ツールの起動

- 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)



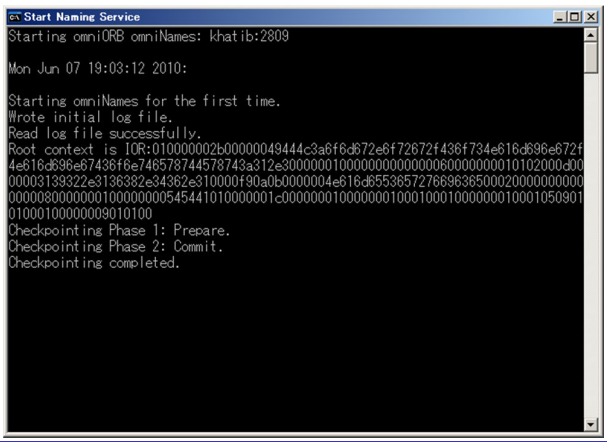
- ワークスペースの切替(通常時)

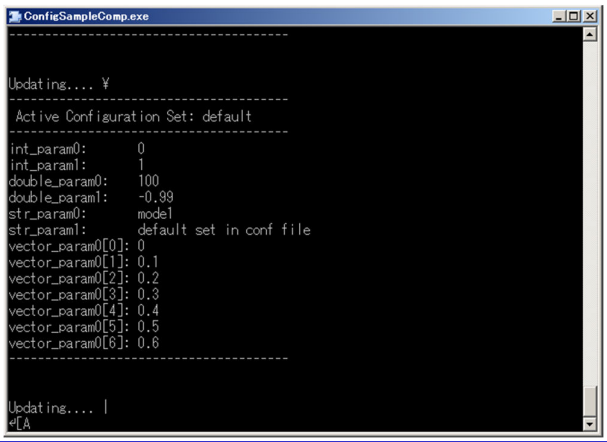


※ワークスペース
Eclipseで開発を行う際の作業領域
Eclipse上でプロジェクトやファイルを作成すると
ワークスペースとして指定したディレクトリ以下に
実際のディレクトリ，ファイルを作成する

準備

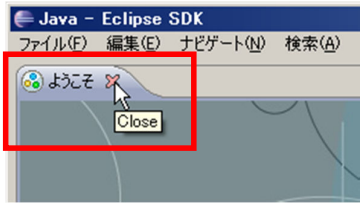
- Naming Serviceの起動
  - [スタート]メニューから
    [プログラム]→[OpenRTM-aist]→[C++]→[tools]→[Start Naming Service]
- ConfigSampleCompの起動
  - [スタート]メニューから起動
    [プログラム]→[OpenRTM-aist]→[C++]
    →[examples]→ [ConfigSampleComp.exe]





準備

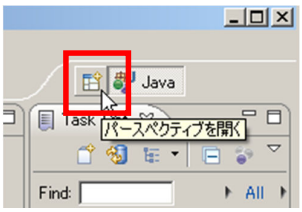
- 初期画面のクローズ
  - 初回起動時のみ

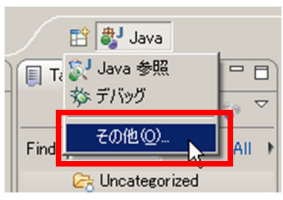


※パースペクティブ
Eclipse上でツールの構成を管理する単位
メニュー，ツールバー，エディタ，ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

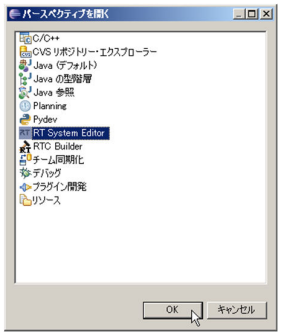
- パースペクティブの切り替え

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



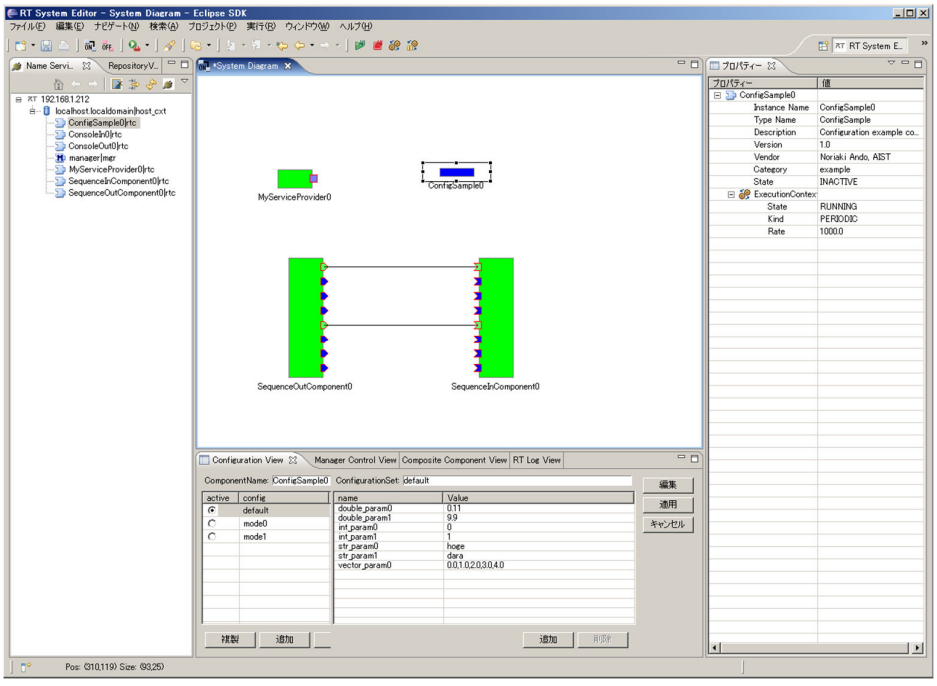
RTSystemEditorについて

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

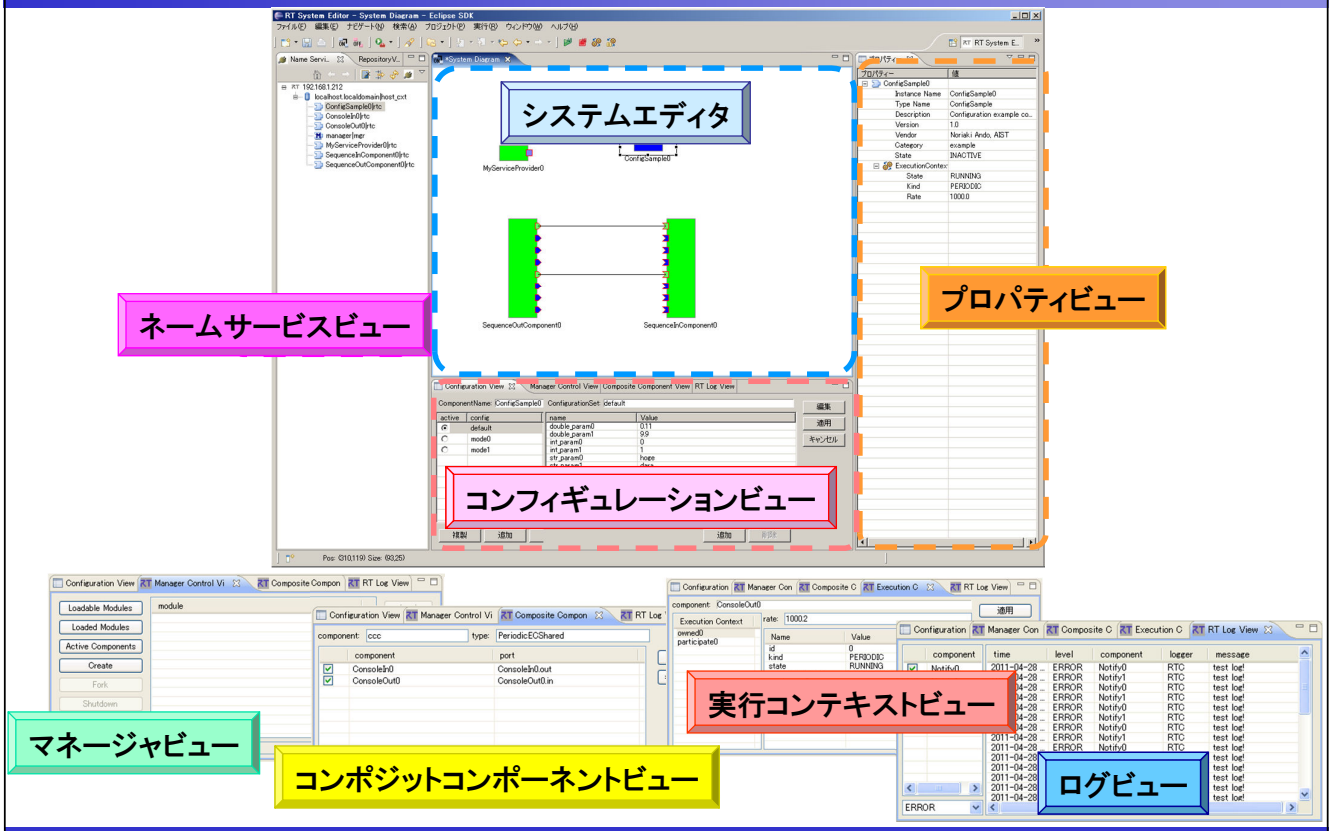
RT MIDDLEWARE

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

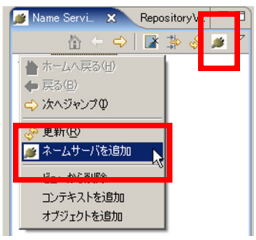


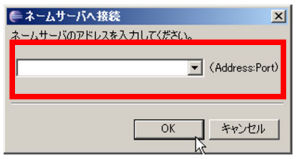
画面構成



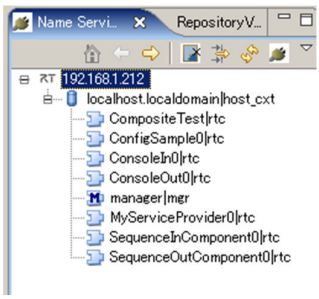
RTシステム構築の基本操作

ネームサービスへ接続

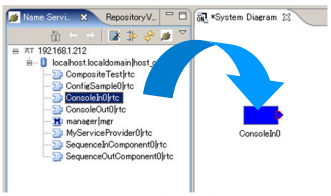




※対象ネームサーバのアドレス，ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能



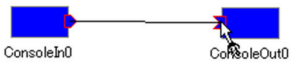
RTコンポーネントの配置



※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

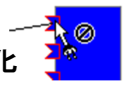
ポートの接続

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



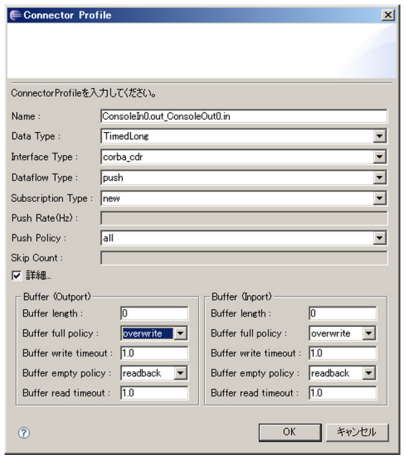
接続プロファイル(DataPort)について

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)



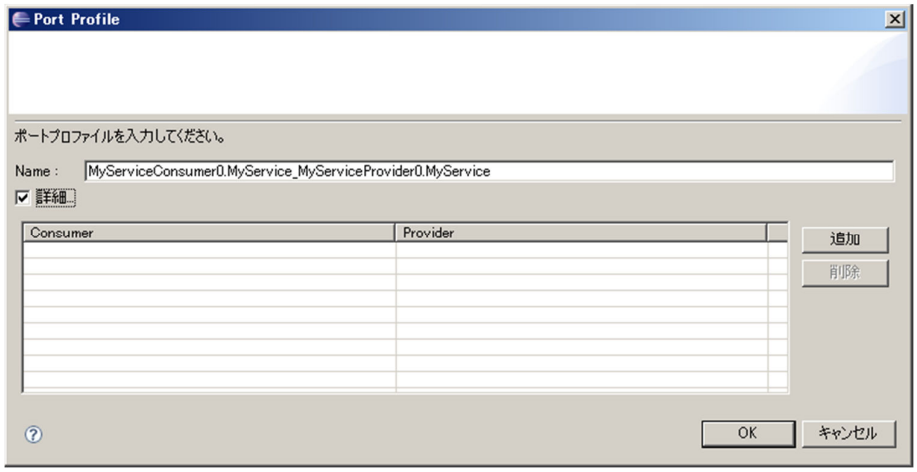
※OutPort側のバッファ, InPort側のバッファそれぞれに設定可能
※timeoutとして「0.0」を設定した場合は, タイムアウトしない

- 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

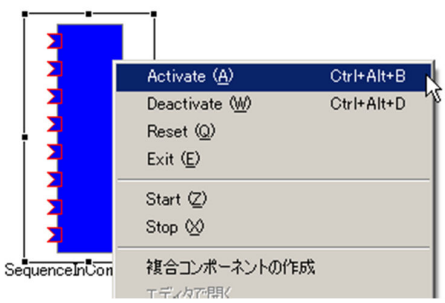
項目	設定内容
Name	接続の名称
インターフェース情報	接続するインターフェースを設定. 接続対象のServicePortに複数のServiceInterfaceが定義されていた場合，どのインターフェースを実際に接続するかを指定



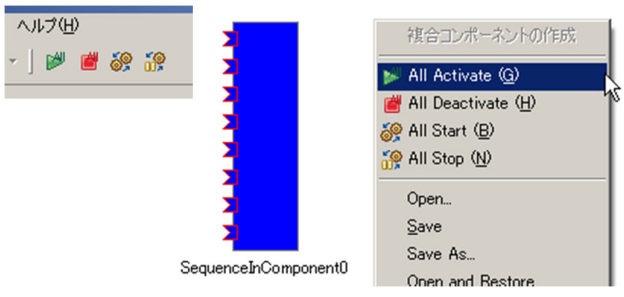
RTコンポーネントの動作

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



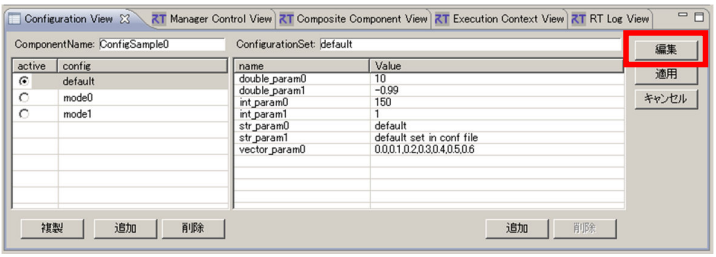
■全コンポーネントの動作を一括変更



※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

コンフィギュレーションビュー

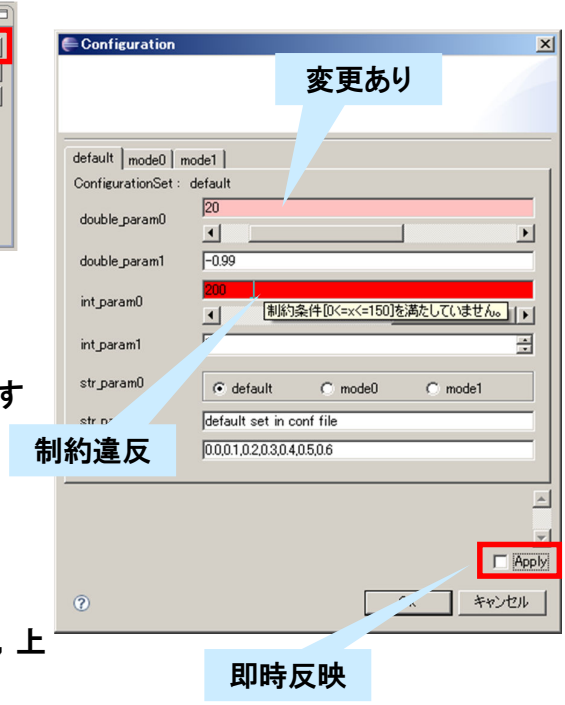
■ RTコンポーネントのコンフィギュレーション情報の確認/編集



※「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を複数保持している場合，上部のタブで編集対象を切り替え



コンフィギュレーション情報の設定方法

- rtc.conf内

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

  ※例) example.ConfigSample.config_file: configsample.conf

- コンフィギュレーションファイル内
  - コンフィギュレーション情報

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

    ※例) conf.mode0.int_param0: 123
  - Widget情報

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

    ※例) conf.__widget__.str_param0: radio
  - 制約情報

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

    ※例) conf.__constraints__.str_param0: (bar,foo,foo,dara)

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

    ※例) conf.__mode1.str_param0: (bar2,foo2,dara2)

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

RTCBuilderを使用することで設定可能

RTCBuilderについて

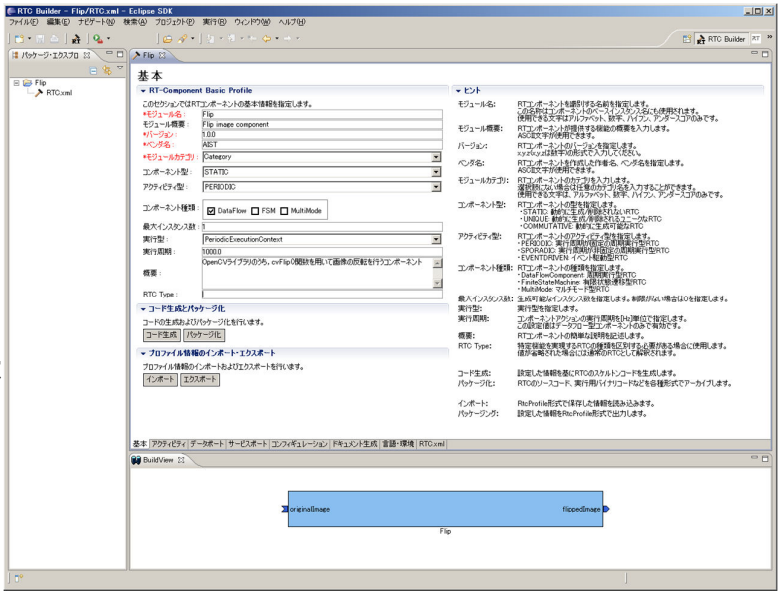
株式会社テクノロジックアート
TECHNOLOGIC ARTS INCORPORATED

RT MIDDLEWARE

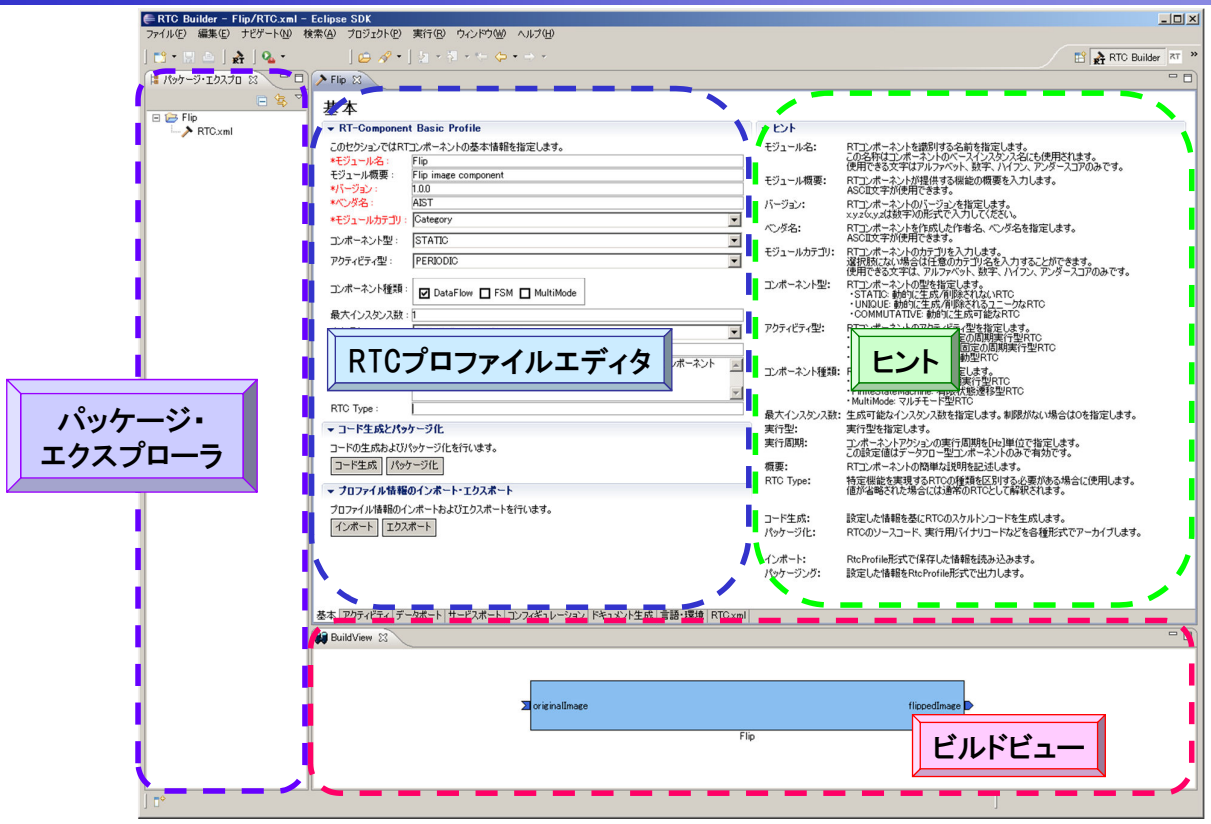
RTCBuilder概要

- 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



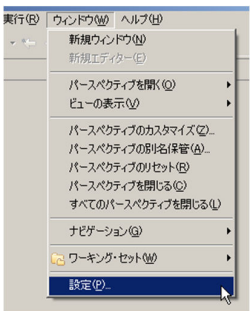
画面構成



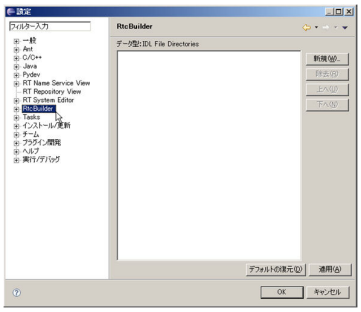
各種設定

- DataPortにて利用するデータ型の指定
  →データ型を定義したIDLファイルが格納されているディレクトリを指定

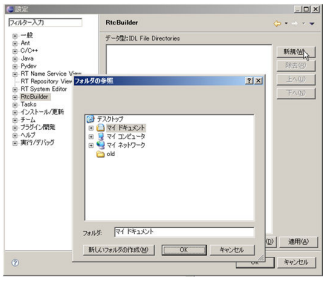
①メニューから
「ウインドウ」ー「設定」



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



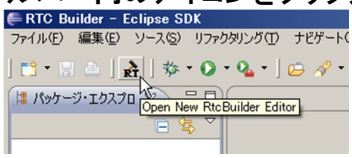
※独自に定義したデータ型を使用する場合のみ必要な設定
OpenRTM-aistにて標準で用意されている型のみを使用する場合には設定不要

・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/

プロジェクト作成/エディタ起動

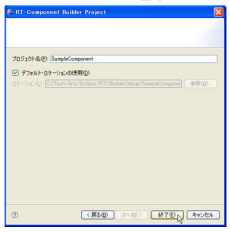
※パースペクティブを「RtcBuilder」に切り替え

①ツールバー内のアイコンをクリック

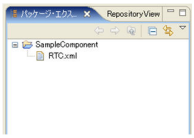


※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し，「次へ」
※メニューから「ファイル」-「Open New Builder Editor」を選択

②「プロジェクト名」欄に入力し，「終了」



③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
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

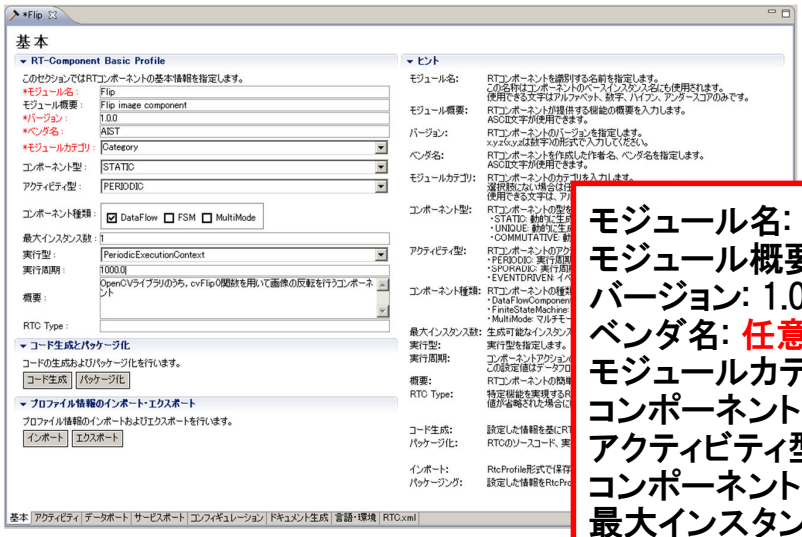
プロジェクト名：Flip

RTCプロファイルエディタ

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

基本プロファイル

- RTコンポーネントの名称など，基本的な情報を設定

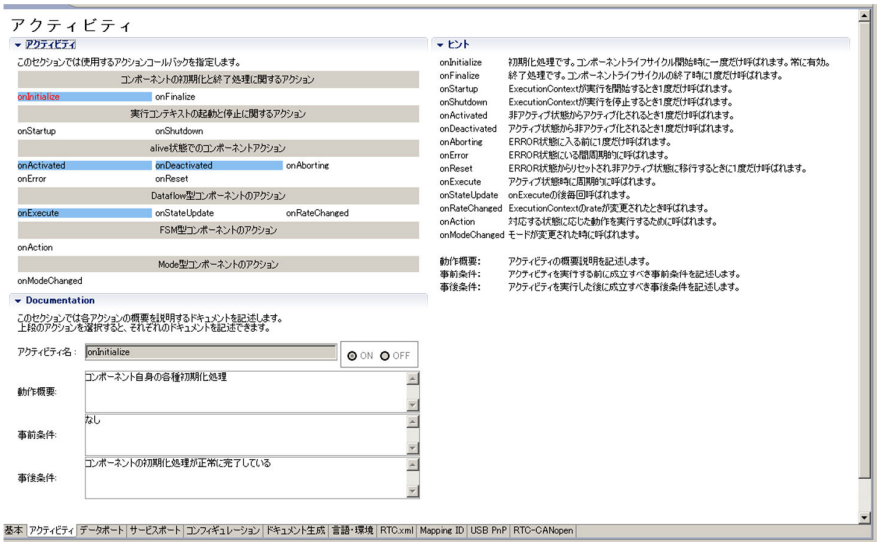


モジュール名: Flip
モジュール概要: 任意(Flip image component)
バージョン: 1.0.0
ベンダ名: 任意(AIST)
モジュールカテゴリ: 任意(Category)
コンポーネント型: STATIC
アクティビティ型: PERIODIC
コンポーネントの種類: DataFlow
最大インスタンス数: 1
実行型: PeriodicExecutionContext
実行周期: 1000.0

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

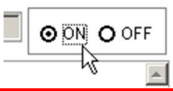
アクティビティ・プロファイル

- 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

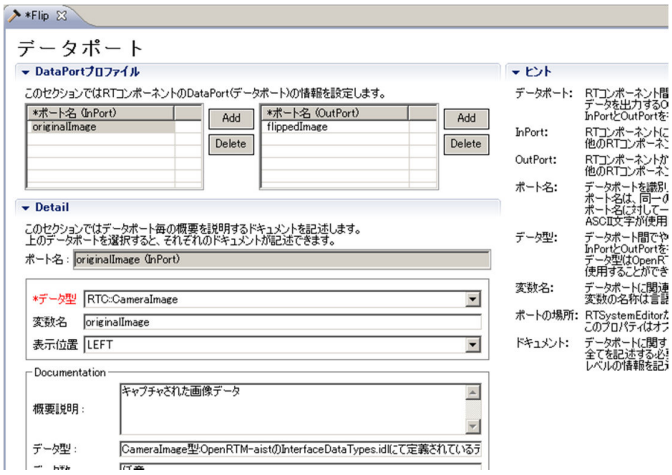


以下をチェック：
onActivated
onDeactivated
onExecute

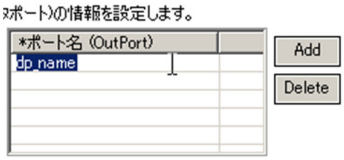
※現在選択中のアクティビティは，一覧画面にて赤字で表示
※使用(ON)が選択されているアクティビティは，一覧画面にて背景を水色で表示
※各アクティビティには，「動作概要」「事前条件」「事後条件」を記述可能
→記述した各種コメントは，生成コード内にDoxygen形式で追加される

データポート・プロファイル

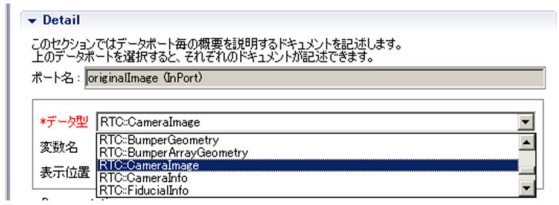
■ 生成対象RTCに付加するDataPortの情報を設定



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



②設定する型情報を一覧から選択



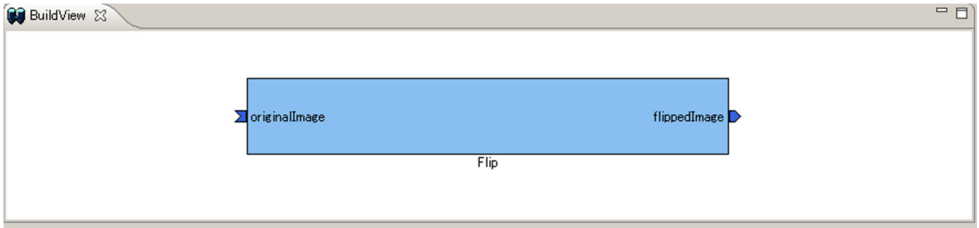
※データ型は，型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

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

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

データポート・プロファイル

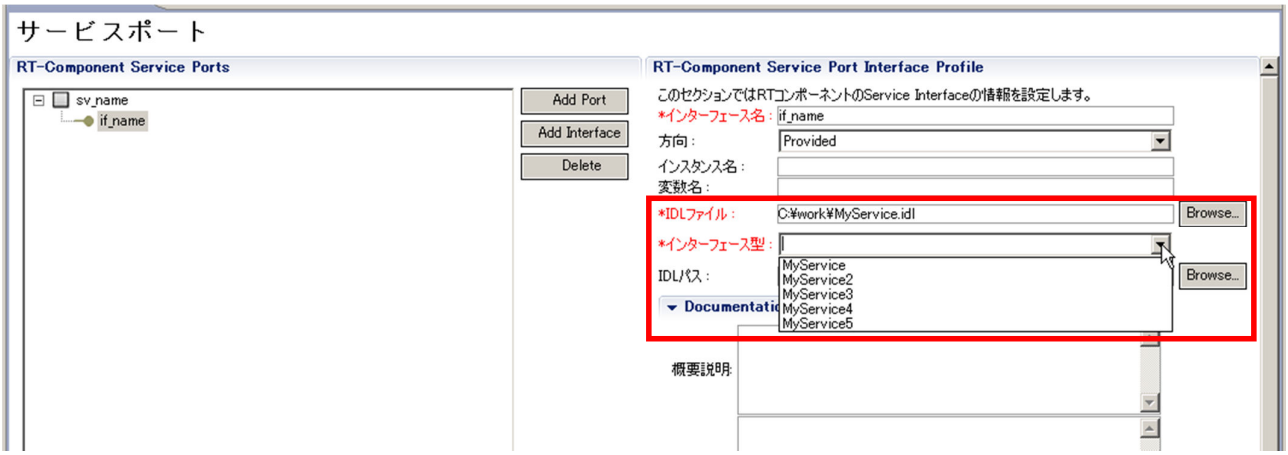
※Portの設定内容に応じて，下部のBuildViewの表示が変化



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

サービスポート・プロファイル

- 生成対象RTCに付加するServicePortの情報を設定



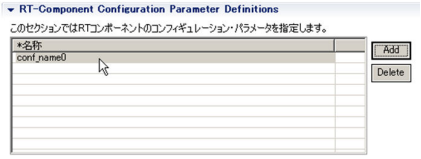
  - サービスインターフェースの指定
    - IDLファイルを指定すると，定義されたインターフェース情報を表示

今回のサンプルでは未使用

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

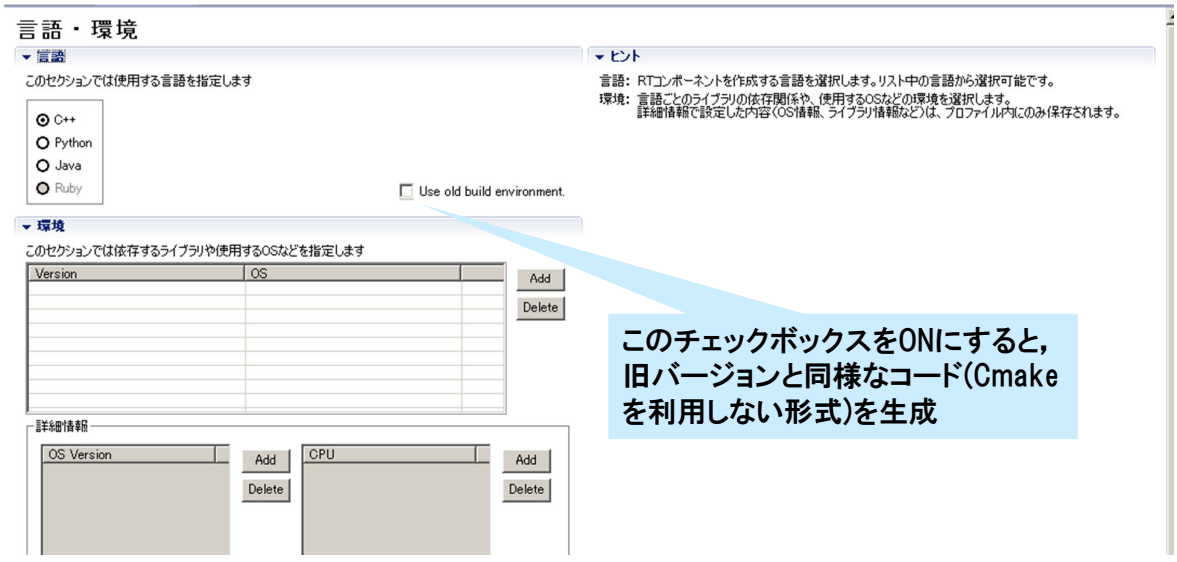
制約条件，Widgetの設定方法

- 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

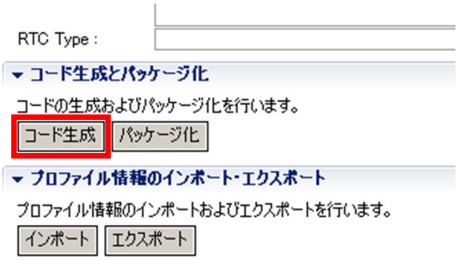


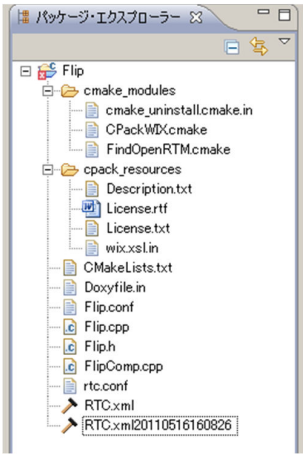
このチェックボックスをONにすると，旧バージョンと同様なコード（Cmakeを利用しない形式）を生成

「C++」を選択

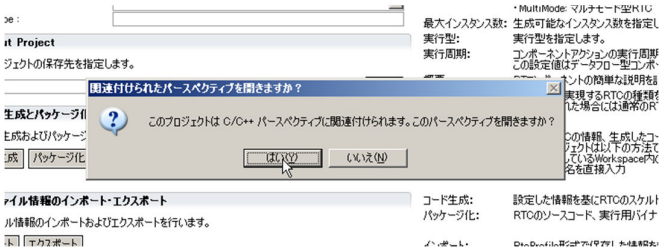
コード生成

- コード生成





- コード生成実行後，パースペクティブを自動切替



※生成コードが表示されない場合には，「リフレッシュ」を実行

C++版RTC → CDT

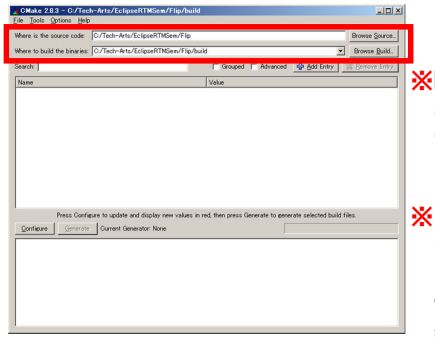
Java版RTC → JDT

（デフォルトインストール済み）

Python版 → PyDev

コンパイル(Windows)

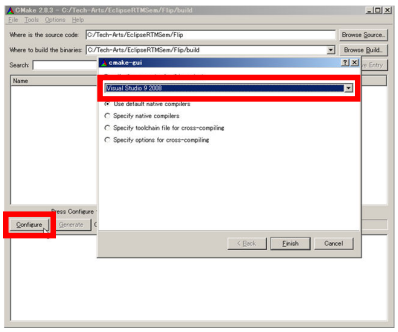
①GUI版Cmakeを起動し，source, binaryのディレクトリを指定



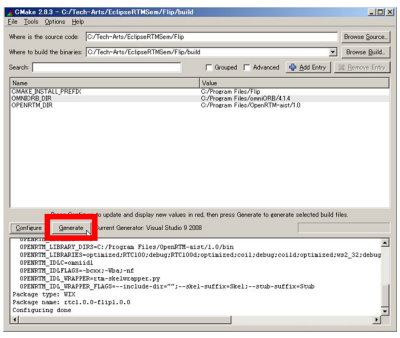
※binaryには，sourceとは別のディレクトリを指定する事を推奨

※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

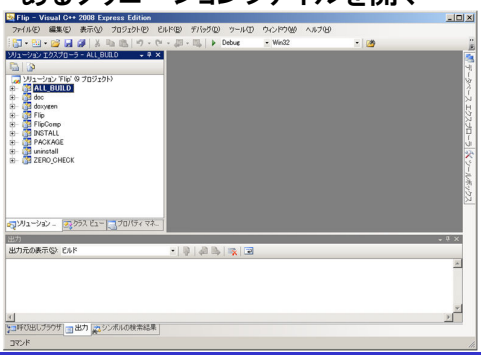
②「Configure」を実行し，使用するプラットフォームを選択

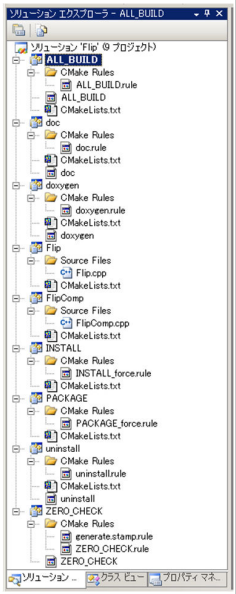


③正常終了後，「Generate」を実行



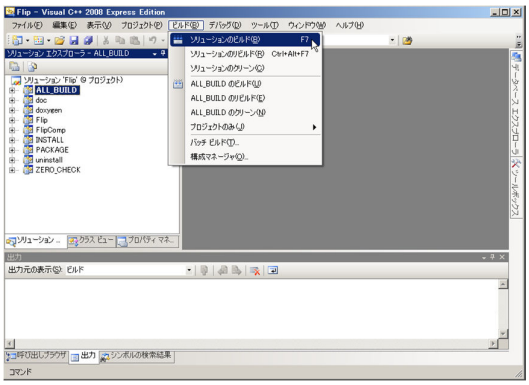
④binaryとして指定したディレクトリ内にあるソリューションファイルを開く



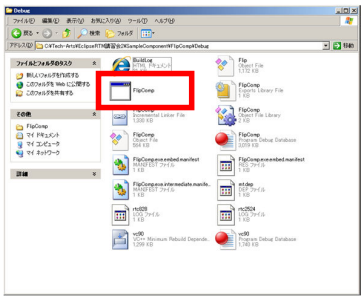


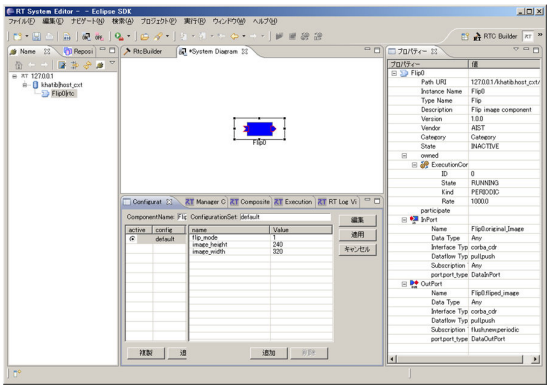
コンパイル・実行(Windows)

⑤ソリューションをビルド



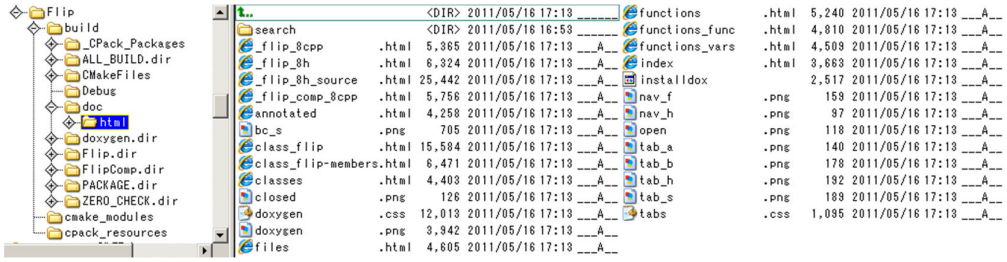
⑥binaryにて指定したディレクトリ以下のDebug内のFlipComp.exeを起動





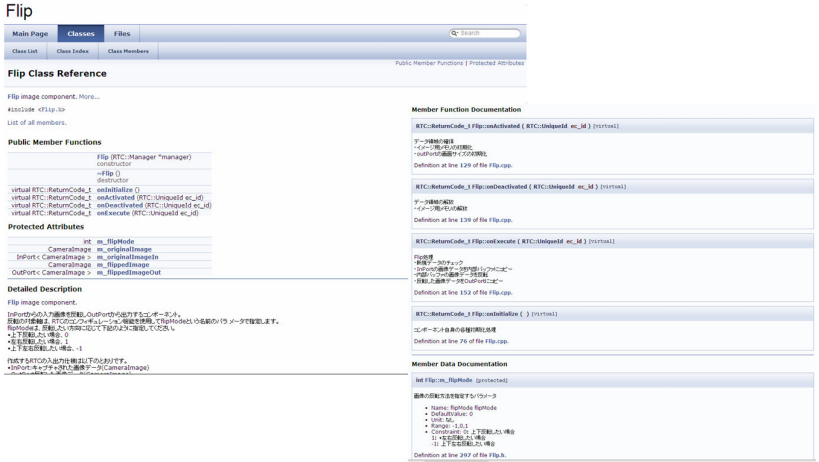
ドキュメント作成(Windows)

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



- **生成されたドキュメントの例**

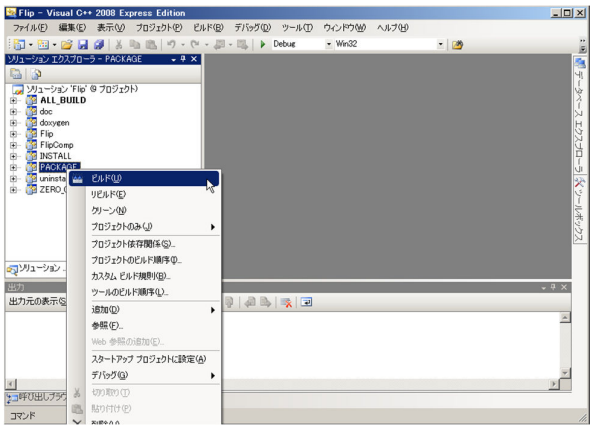
Flip



Flip

配布用パッケージ作成(Windows)

- ソリューション中の「PACKAGE」をビルド



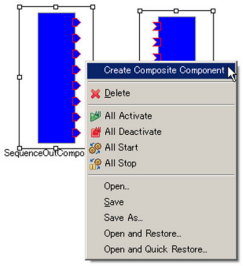
- binaryにて指定したディレクトリ直下にmsi形式のインストールパッケージを生成
  - コンポーネントのインストール先
    C:/Program Files/OpenRTM-aist/1.1/components/<言語>/<パッケージ名>

RTSystemEditorの補足

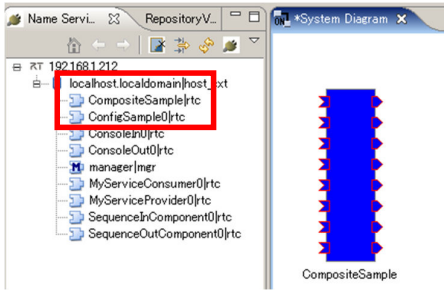
複合コンポーネント

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

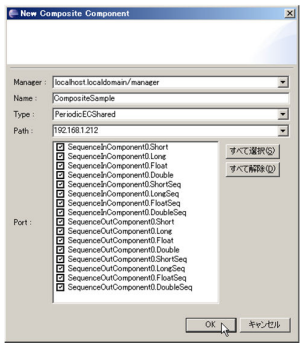
①複数RTCを選択している状態で右クリック



③複合コンポーネントを生成



②複合コンポーネントのプロパティを設定



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

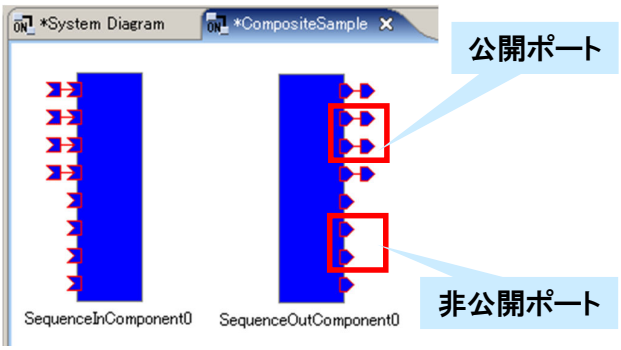
※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

複合コンポーネント

- 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



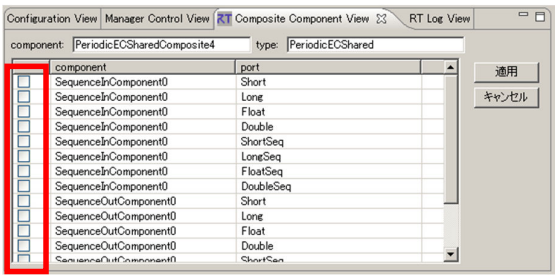
※エディタ内に別RTCをDnDすることで，子コンポーネントの追加が可能
→追加したRTCのポートは全て非公開に設定
※エディタ内のRTCを削除することで，子コンポーネントの削除が可能
→削除されたRTCは，親エディタに表示

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

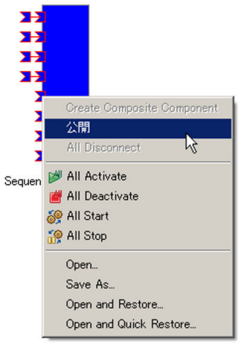
ポート公開情報



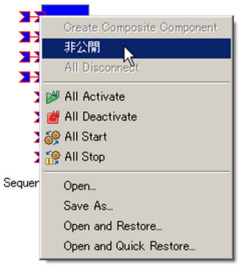
※ポート公開情報を変更し，「適用」をクリック

- 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」



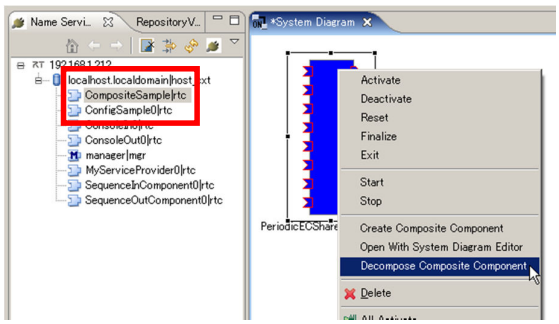
外部コンポーネントと接続されているポートを「非公開」に設定することはできません

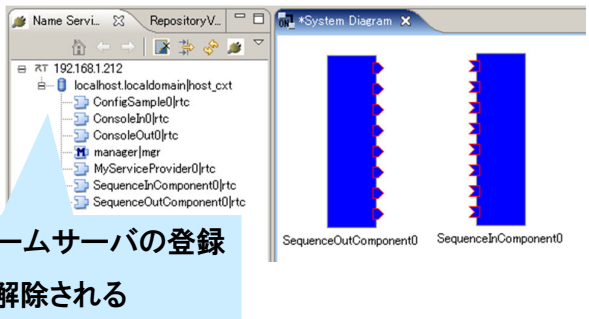
複合コンポーネント

■ 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



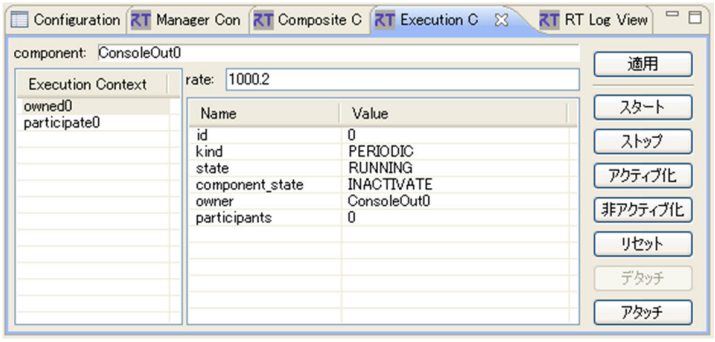


ネームサーバの登録も解除される

※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

実行コンテキストビュー

- RTコンポーネントが属する実行コンテキスト(EC)を一覧表示

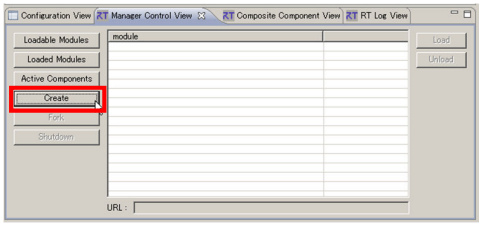


属性名	説明
id	ECのID．オンラインの場合には，context_handleを表示
kind	ECの種別(PERIODIC/EVENT_DRIVEN/OTHER)
state	ECの状態(RUNNING/STOPPING)
component state	対象RTCの状態(ACTIVE/INACTIVE/ERROR)
owner	対象ECを所有しているオーナーRTCのインスタンス名
participants	対象ECに参加中のRTCの数

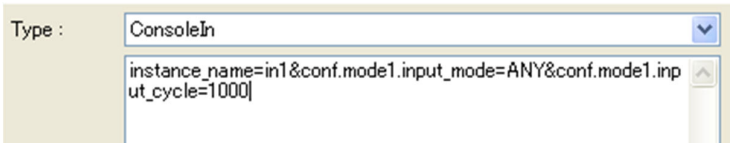
※対象ECの実行周期の変更，EC自身の動作開始/終了，新規RTCへのアタッチ，アタッチ済みRTCのデタッチも可能

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

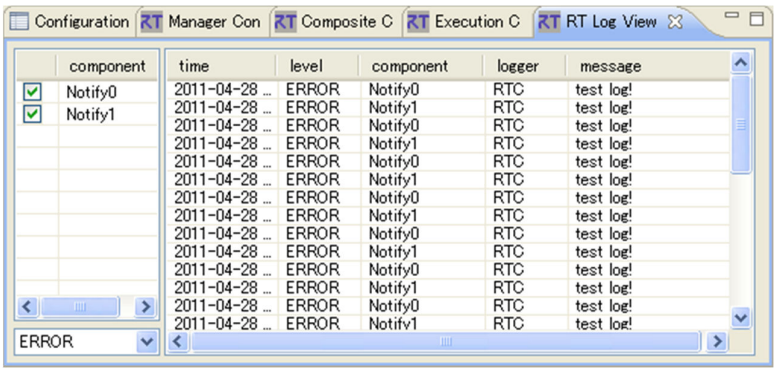


- コンフィギュレーション指定パラメータ
  - conf.［ConfigSet名］.［Configパラメータ名］=［設定値］の形式にてConfigurationSetの値も設定可能



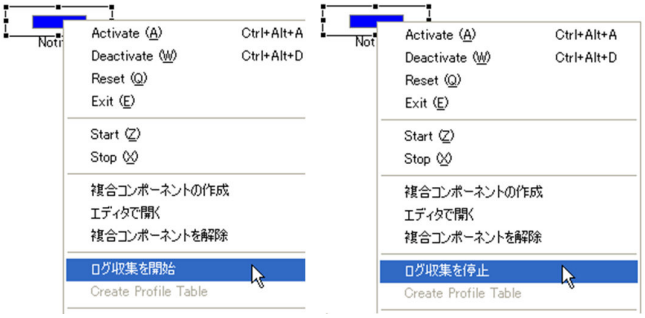
ログビュー

- 選択したRTCから収集したログ情報を一覧表示

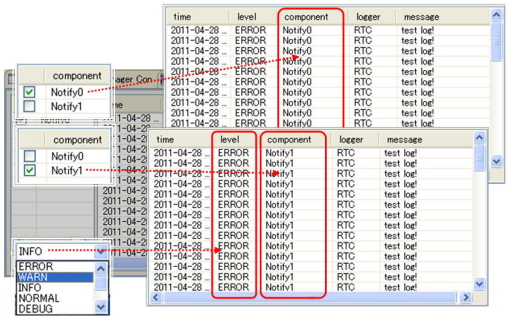


※近日機能追加予定

- ログ収集の開始/停止

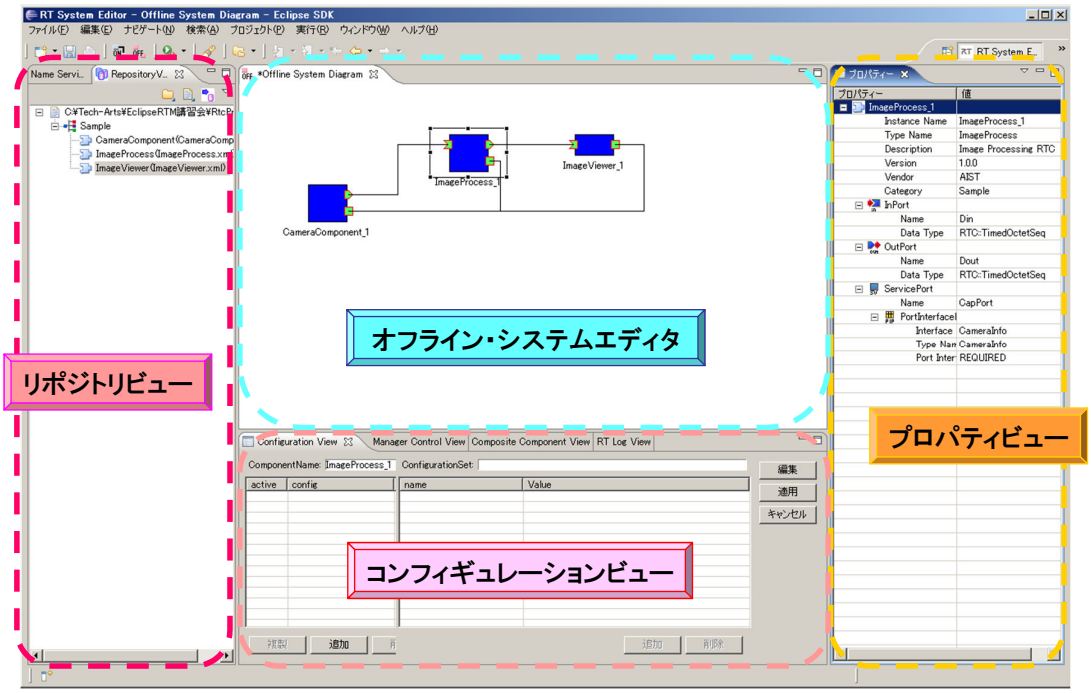


- ログ情報のフィルタリング



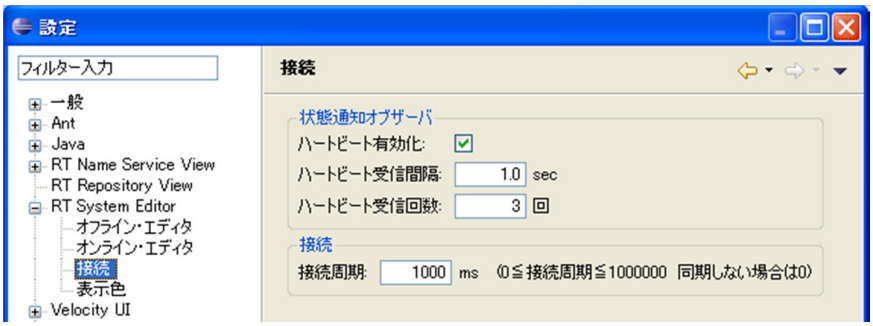
オフラインエディタ

- 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

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



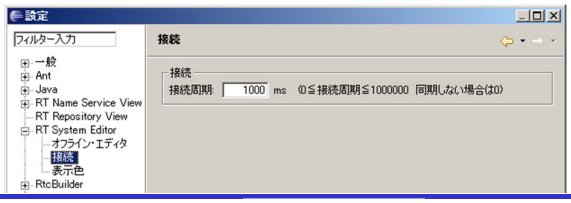
  - ハートビート有効化：ハートビートによる生存確認機能の有効化
  - ハートビート受信間隔：ハートビートの受信間隔．この間隔以内にRTC側からハートビートが送られてこないと生存確認失敗と判断
  - ハートビート受信回数：この回数を超えて生存確認に失敗した場合，対象RTCに異常が発生したと判断

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

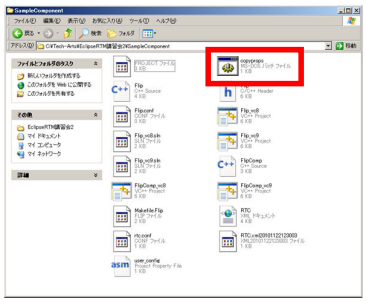
RTCBuilderの補足

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

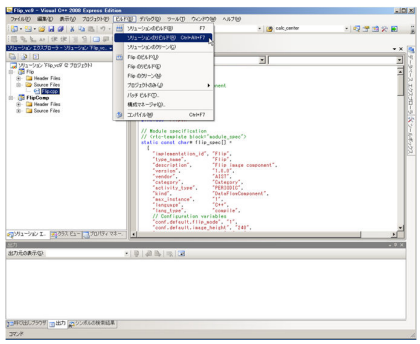
RT MIDDLEWARE

コンパイル・実行(Cmake未使用)

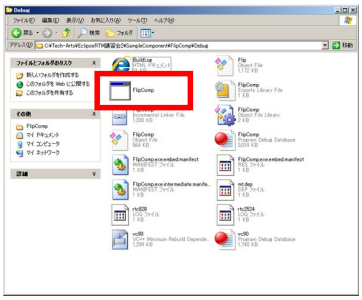
①コード生成先ディレクトリ内の
「copyprops.bat」をダブルクリックして，
設定ファイルをコピー

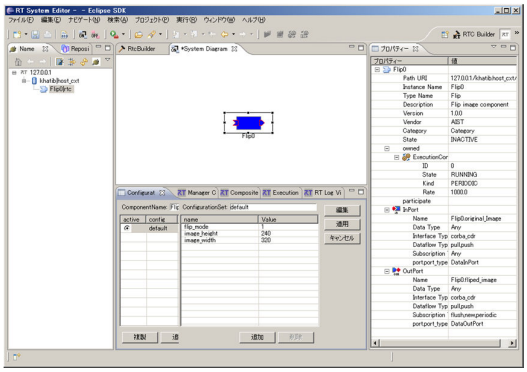


②VisualStudioを用いたビルド



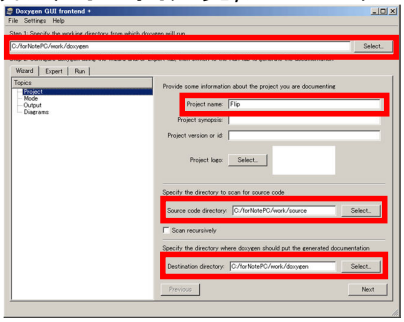
③FlipComp/Debug内のFlipComp.exeを起動



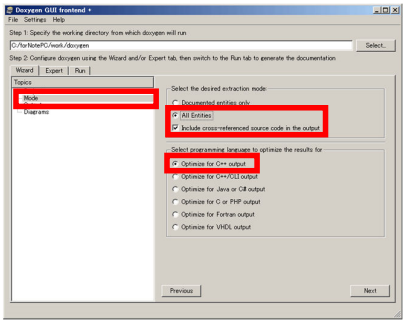


ドキュメント作成

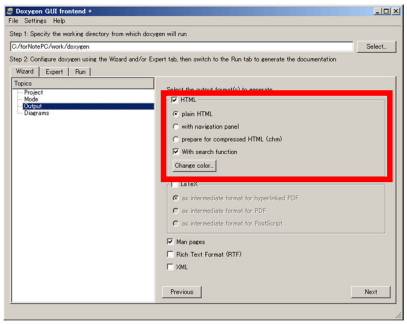
①Doxygen用GUIツールを起動
作業用ディレクトリ,ソース格納場所,
生成ファイル出力先,プロジェクト名を指定



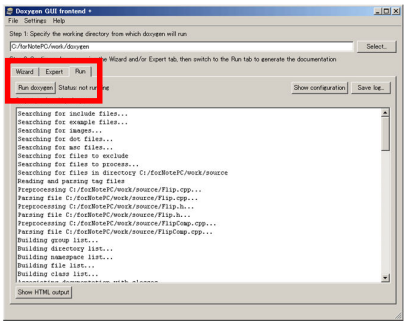
②「Mode」セクションにて,
出力内容,使用言語を指定



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



RTミドルウエア(OpenRTM-aist-1.1.0)講習会