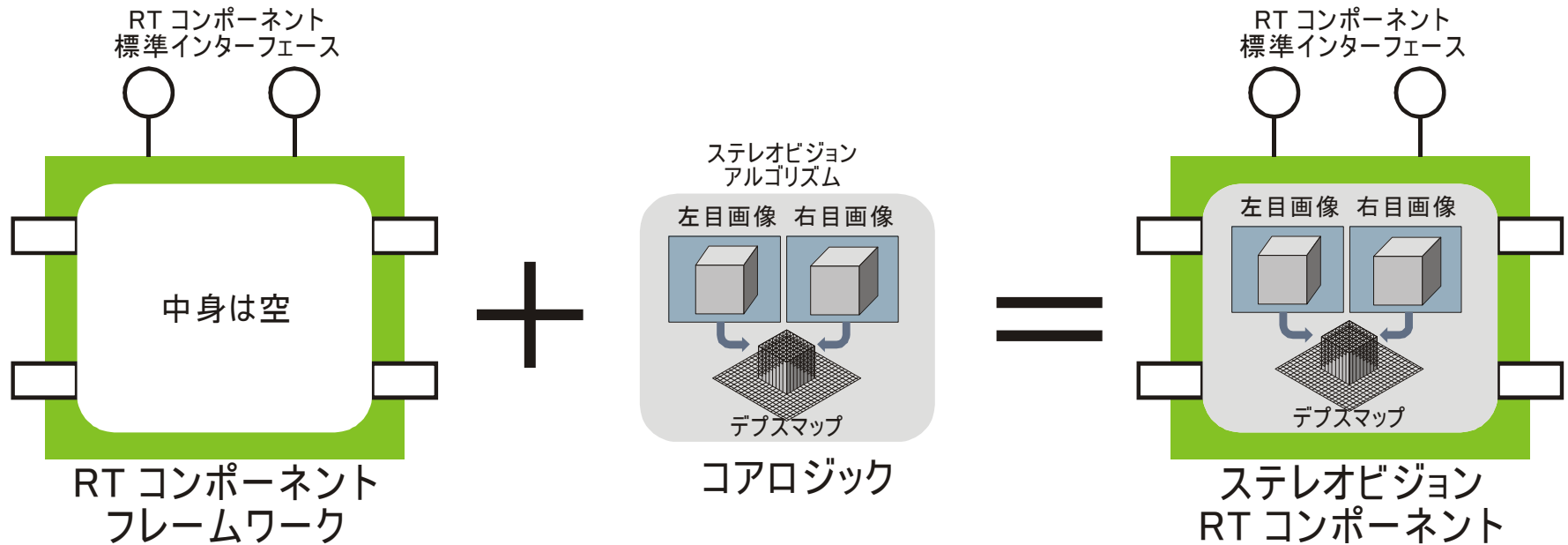


RTコンポーネントの開発

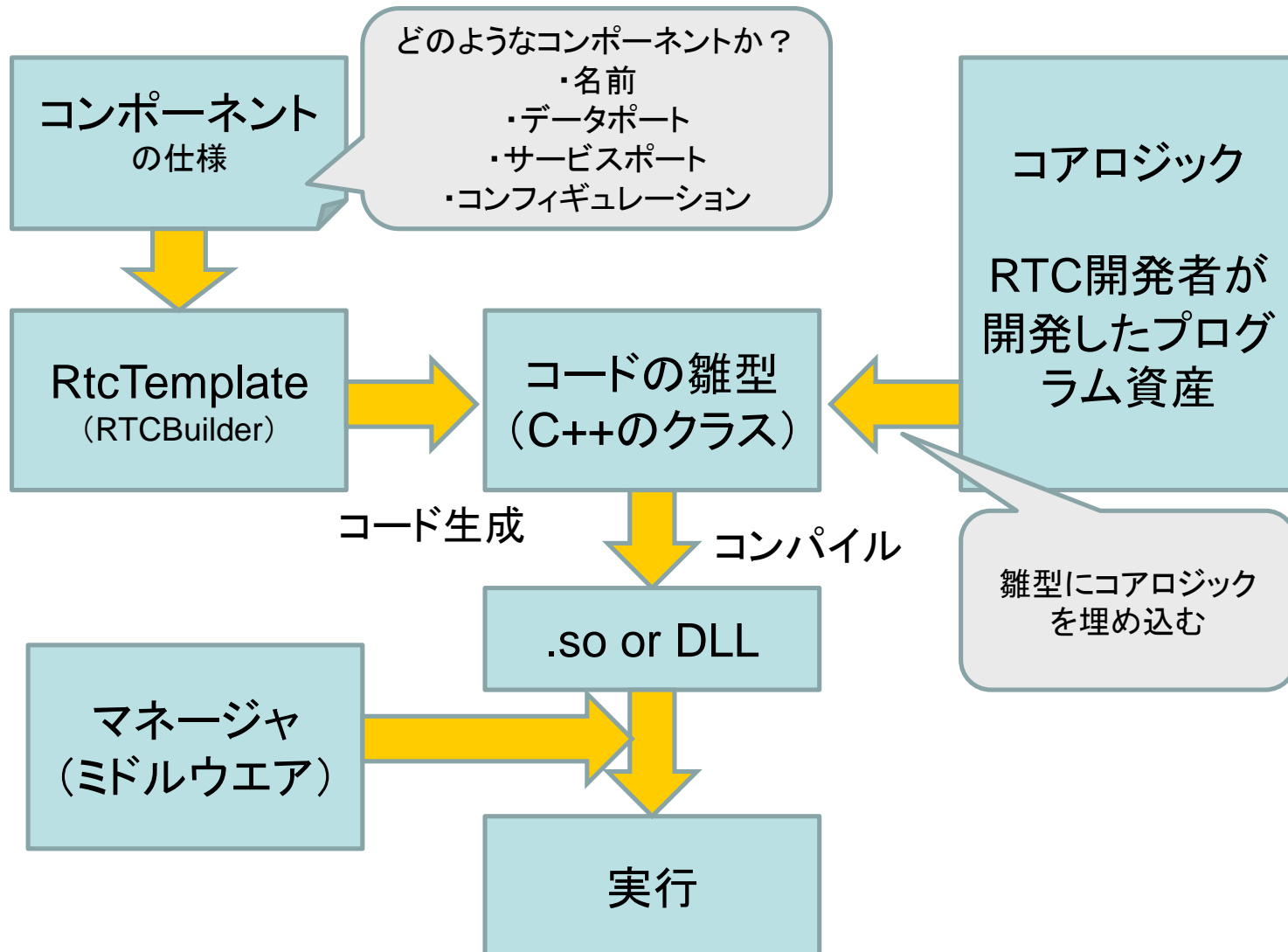


フレームワークとコアロジック

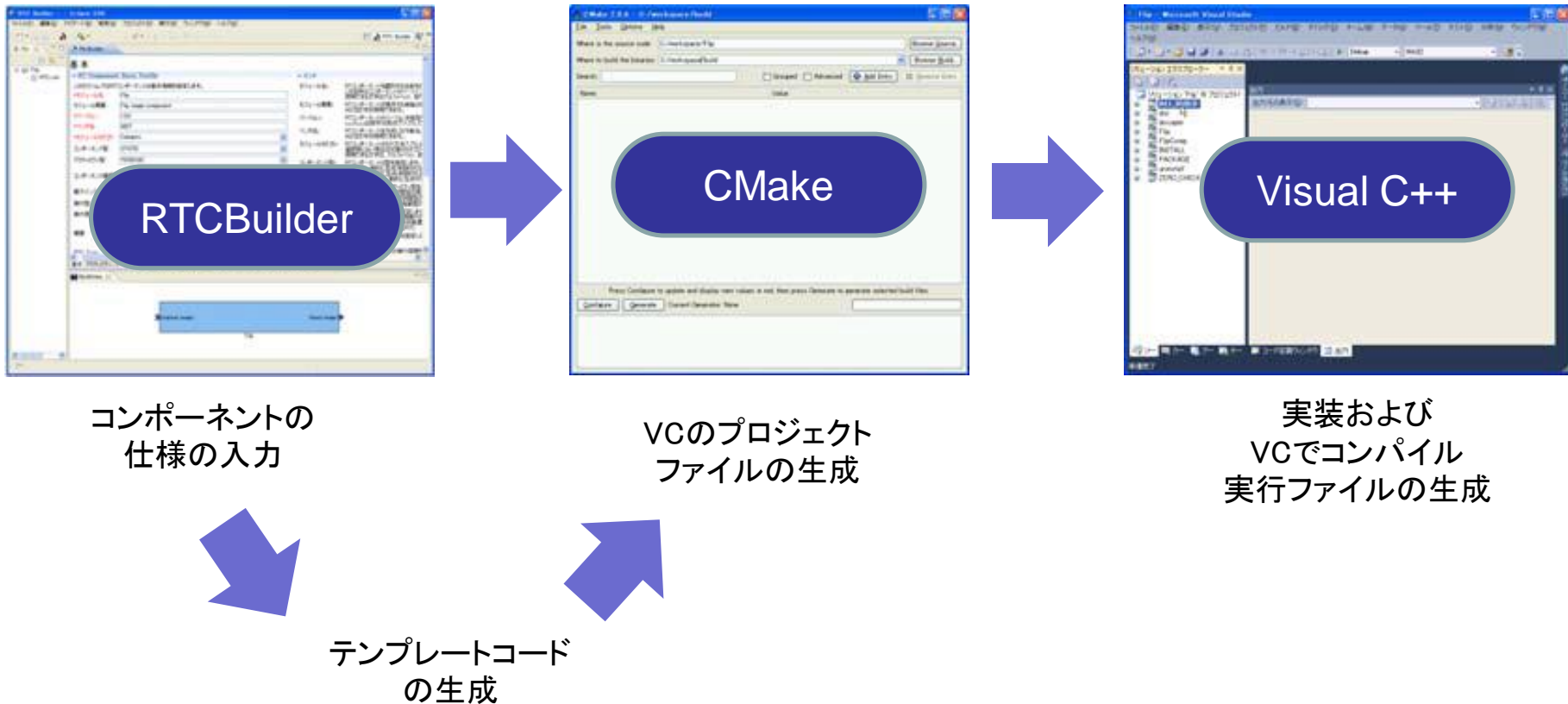


RTCフレームワーク+コアロジック=RTコンポーネント

OpenRTMを使った開発の流れ

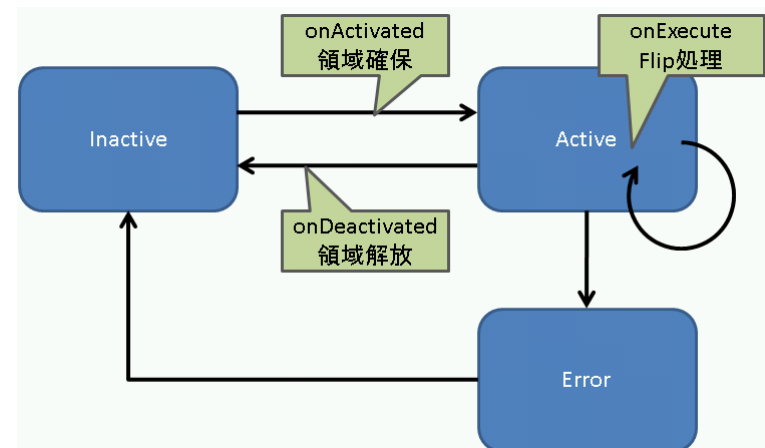


コンポーネントの作成 (Windowsの場合)



Flipコンポーネントの実装

- CMakeで生成したソリューションファイルをVisual C++で開く
- Flip.cppの実装
 - onInitialize: コンポーネントの初期化
 - onActivate: データ保存領域の確保
 - onDeactivate: データ保存領域の解放
 - onExecute: Flip処理



コアロジックの実装

- 生成されたクラスのメンバー関数に必要な処理を記述
- 主要な関数
 - onExecute (周期実行)
- 処理
 - InPortから読む
 - OutPortへ書く
 - サービスを呼ぶ
 - コンフィギュレーションを読む

```
class MyComponent
: public DataflowComponentBase
{
public:
    // 初期化時に実行したい処理
    virtual ReturnCode_t onInitialize()
    {
        if (mylogic.init())
            return RTC::RTC_OK;
        return RTC::RTC_ERROR;
    }

    // 周期的に実行したい処理
    virtual ReturnCode_t onExecute(RTC::UniqueId ec_id)
    {
        if (mylogic.do_something())
            return RTC::RTC_OK;
        return RTC::RTC_ERROR;
    }

private:
    MyLogic mylogic;
    // ポート等の宣言
    //   :
};
```

コールバック関数

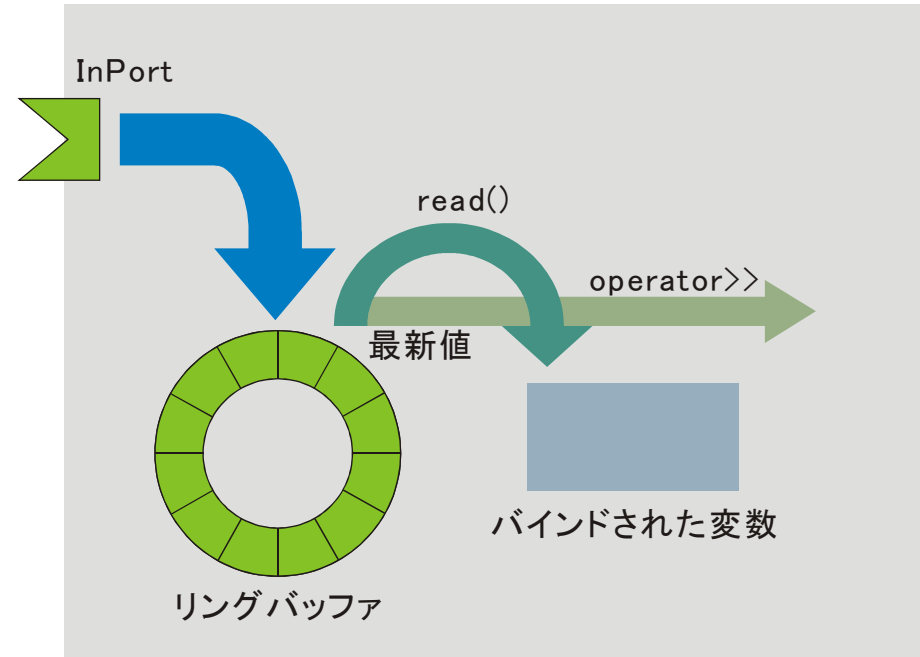
RTCの作成＝コールバック関数に処理を埋め込む

コールバック関数	処理
onInitialize	初期化処理
onActivated	アクティブ化されるとき1度だけ呼ばれる
onExecute	アクティブ状態時に周期的に呼ばれる
onDeactivated	非アクティブ化されるとき1度だけ呼ばれる
onAborting	ERROR状態に入る前に1度だけ呼ばれる
onReset	resetされる時に1度だけ呼ばれる
onError	ERROR状態のときに周期的に呼ばれる
onFinalize	終了時に1度だけ呼ばれる
onStateUpdate	onExecuteの後毎回呼ばれる
onRateChanged	ExecutionContextのrateが変更されたとき1度だけ呼ばれる
onStartup	ExecutionContextが実行を開始するとき1度だけ呼ばれる
onShutdown	ExecutionContextが実行を停止するとき1度だけ呼ばれる

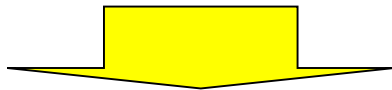
とりあえずは
この5つの関数
を押さえて
おけばOK

InPort

- InPortのテンプレート第2引数: バッファ
 - ユーザ定義のバッファが利用可能
- InPortのメソッド
 - read(): InPort バッファからバインドされた変数へ最新値を読み込む
 - >>: ある変数へ最新値を読み込む

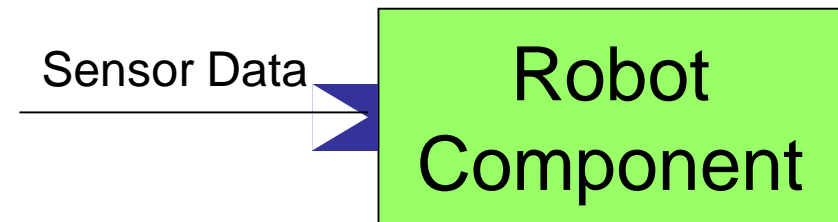


基本的にOutPortと対になる



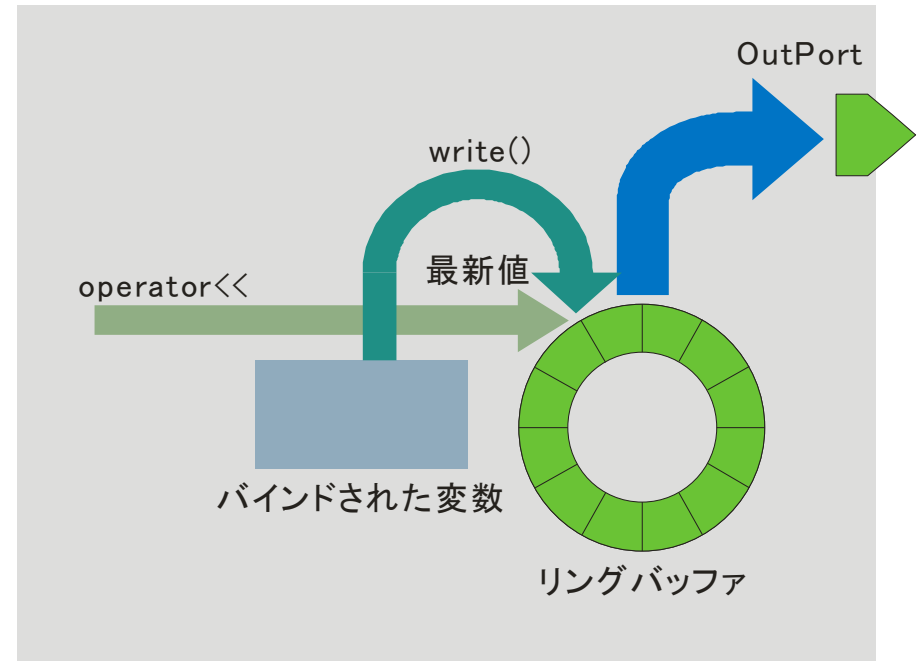
データポートの型を
同じにする必要あり

例

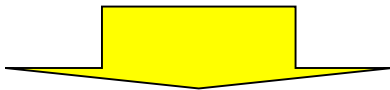


OutPort

- OutPortのテンプレート第2引数:
バッファ
 - ユーザ定義のバッファが利用可能
- OutPortのメソッド
 - write(): OutPort バッファへ
バインドされた変数の最新値
として書き込む
 - >>: ある変数の内容を最新
値としてリングバッファに書き
込む

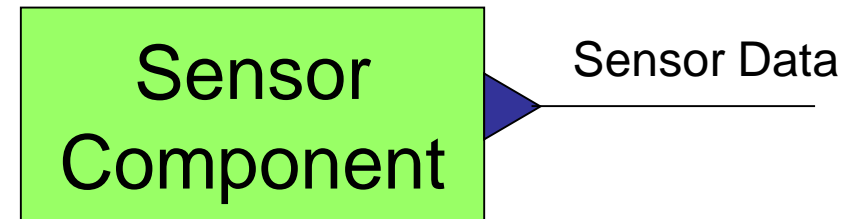


基本的にInPortと対になる



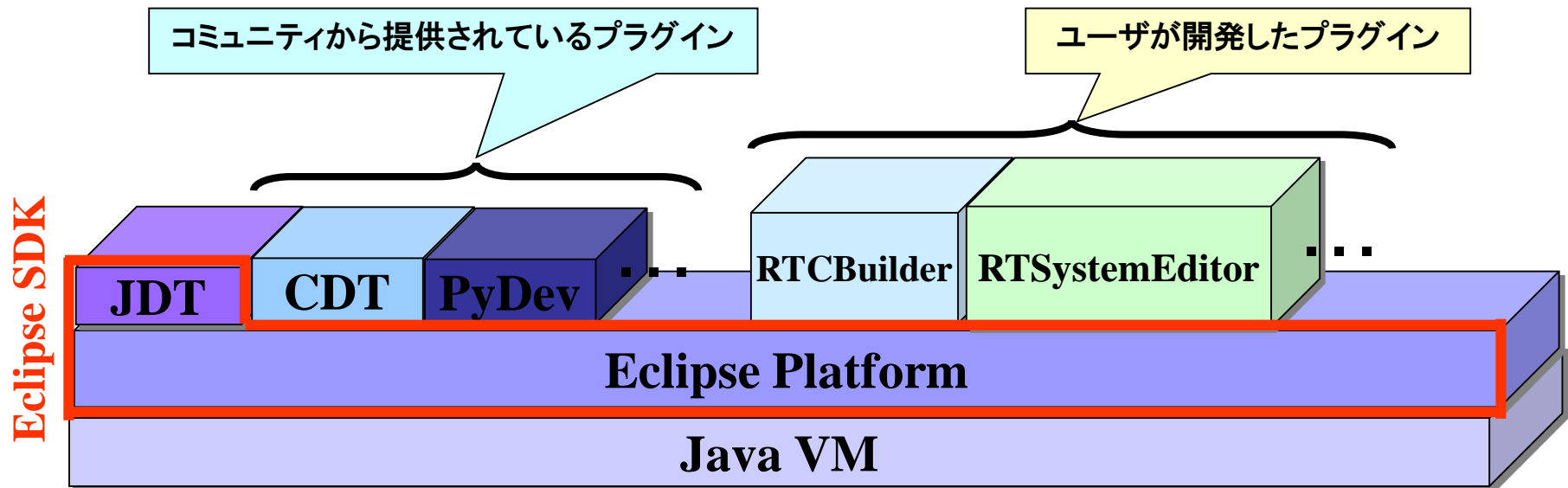
データポートの型を
同じにする必要あり

例



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



システム構築支援ツール RTSystemEditorについて

- 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

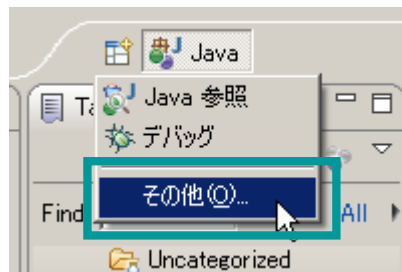
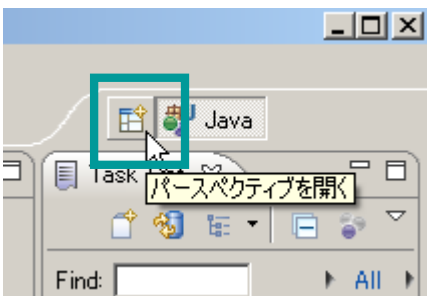
The screenshot displays the RTSystemEditor interface with several views and components highlighted by colored boxes and labels:

- システムエディタ** (System Editor): The central workspace showing a diagram with components like `MyServiceProvider`, `ConfigSample0`, and `SequenceOutComponent0`.
- ネームサービスビュー** (Name Service View): A view on the left showing a tree structure of components and their relationships.
- プロパティビュー** (Property View): A view on the right showing the properties of the selected component, such as `ConfigSample0`.
- コンフィギュレーションビュー** (Configuration View): A view at the bottom showing the configuration of the selected component, including parameters like `double_param0` and `int_param0`.
- マネージャビュー** (Manager View): A view on the bottom left showing the loaded modules and active components.
- 複合コンポーネントビュー** (Composite Component View): A view at the bottom center showing the configuration of the selected composite component, including its type and parameters.
- 実行コンテキストビュー** (Execution Context View): A view on the bottom right showing the execution context of the selected component, including its rate and kind.
- ログビュー** (Log View): A view on the bottom right showing the log messages of the selected component, including the level, component, logger, and message.

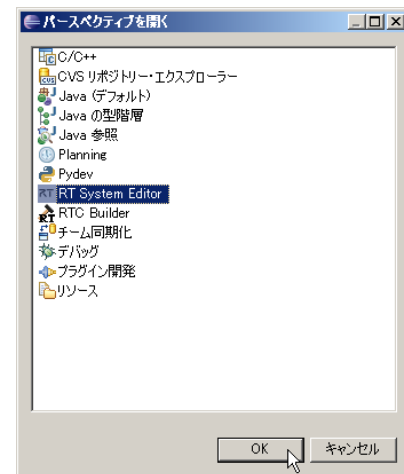
- Naming Serviceの起動
 - [スタート]メニューから
 - [プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]
- CameraViewerCompの起動
 - [スタート]メニューから起動
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
→[opencv-rtcs]→ [CameraViewerComp.exe]
- DirectShowCamCompの起動
 - [スタート]メニューから起動
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]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

n 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



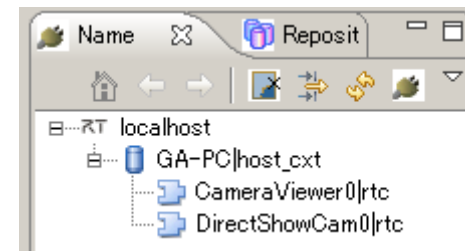
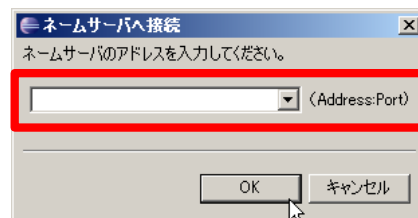
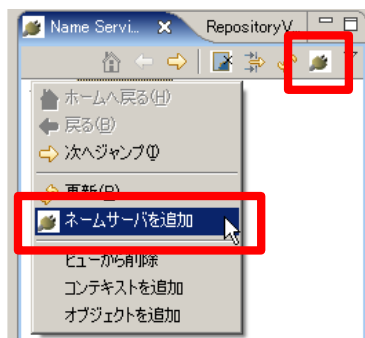
②一覧画面から対象ツールを選択



※パースペクティブ

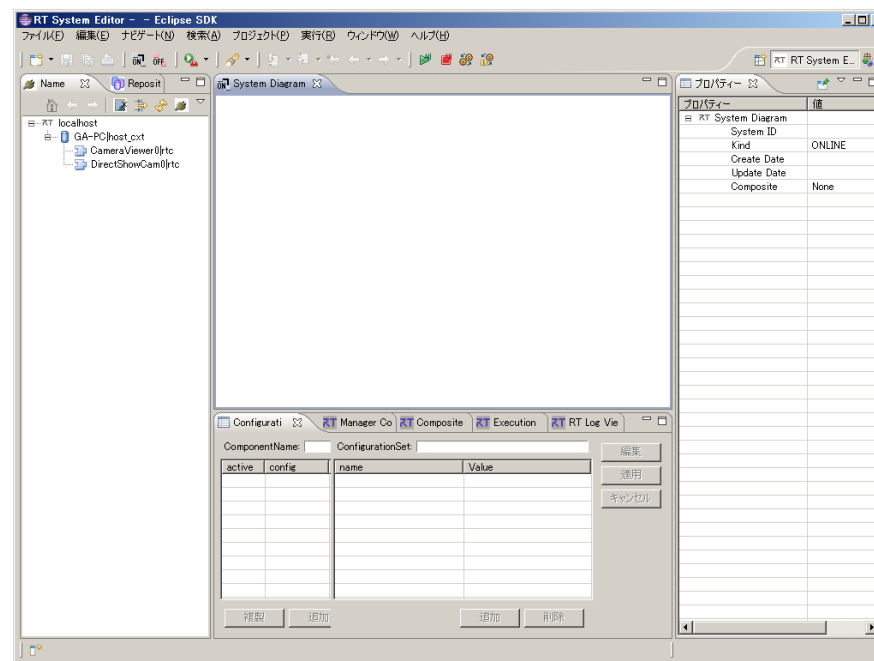
Eclipse上でツールの構成を管理する単位
メニュー、ツールバー、エディタ、ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

● ネームサービスへ接続

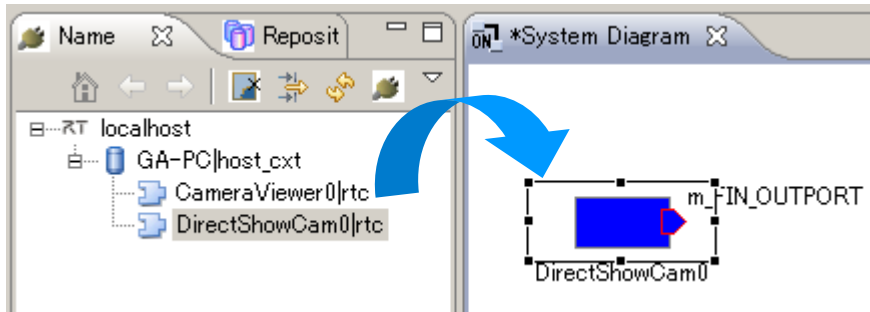


※対象ネームサーバのアドレス、ポートを指定
→ポート省略時のポート番号は
設定画面にて設定可能

● システムエディタの起動



● RTコンポーネントの配置



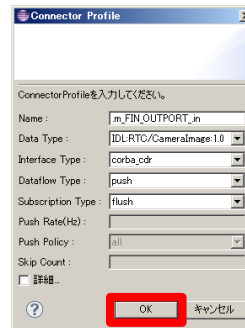
※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

● ポートの接続

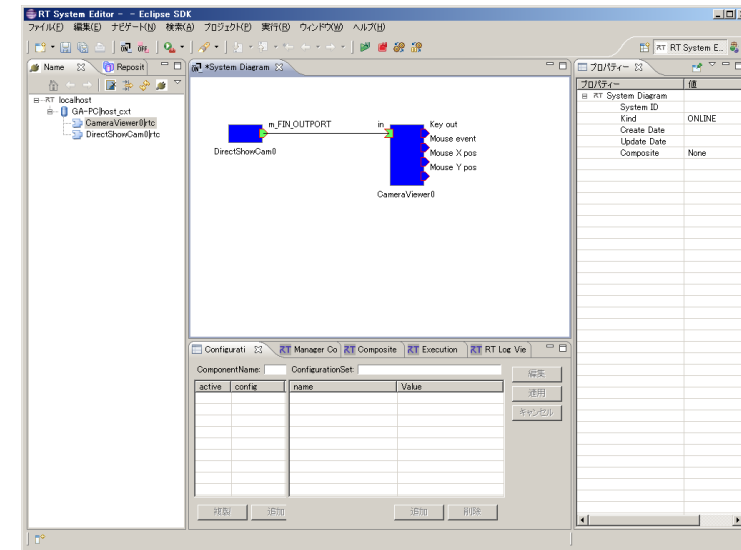
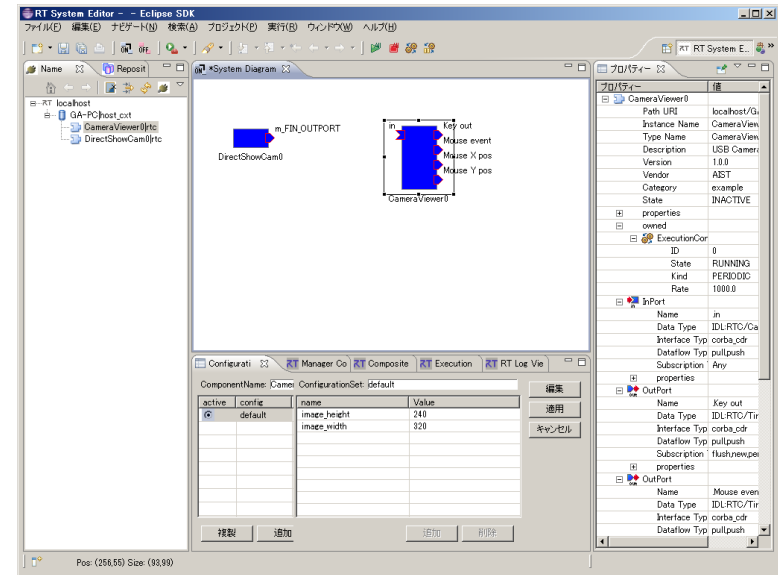
①接続元のポートから接続先のポートまでドラッグ



②【接続プロファイル】画面で「OK」を選択

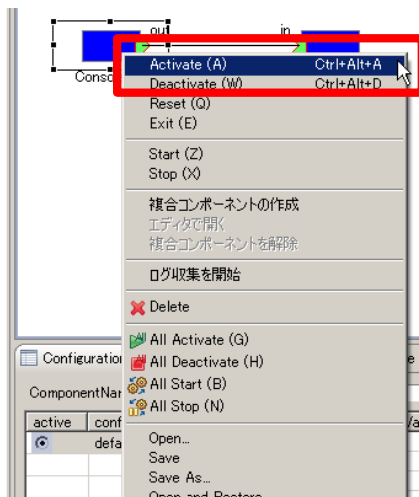


※ポートのプロパティが異なる場合など、接続不可能なポートの場合にはアイコンが変化

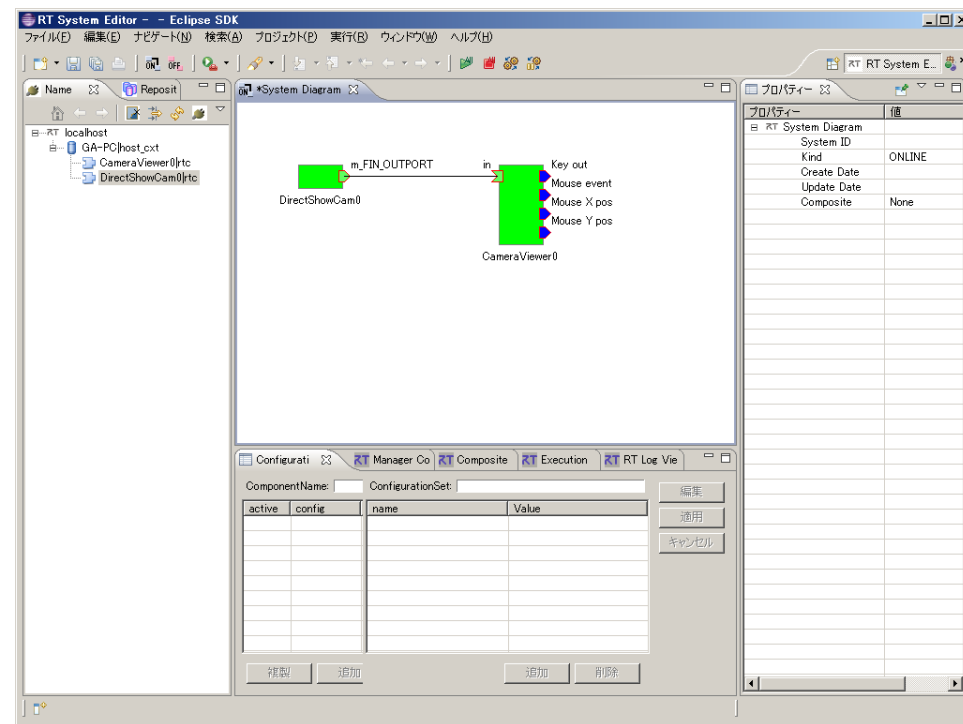
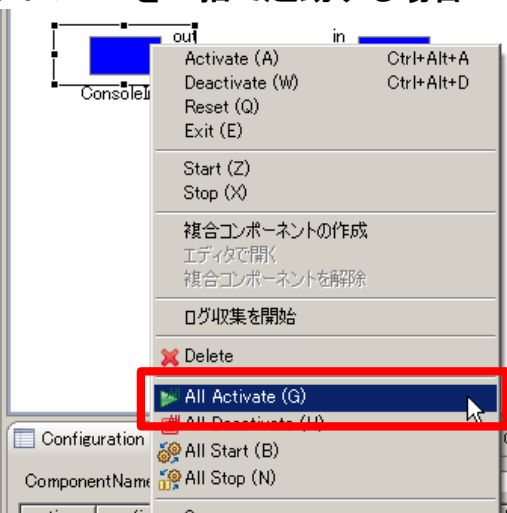


● コンポーネントの起動

※各RTC単位で起動する場合



※全てのRTCを一括で起動する場合



※停止はDeactivateを実行

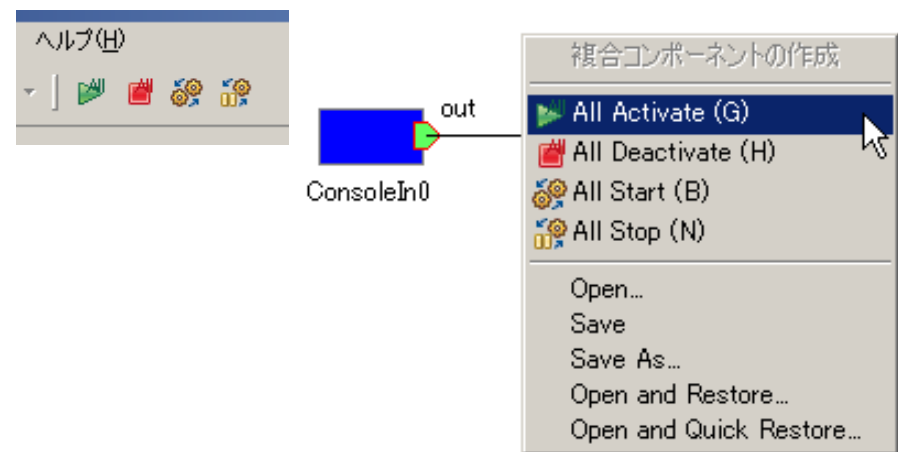
※RTC間の接続を切る場合には接続線をDelete
もしくは、右クリックメニューから「Delete」を選択

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

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

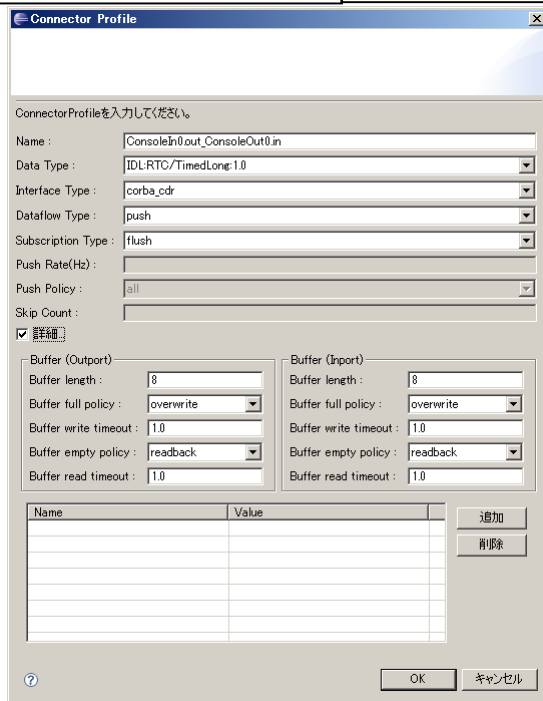
n SubscriptionType

- 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

n 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)

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

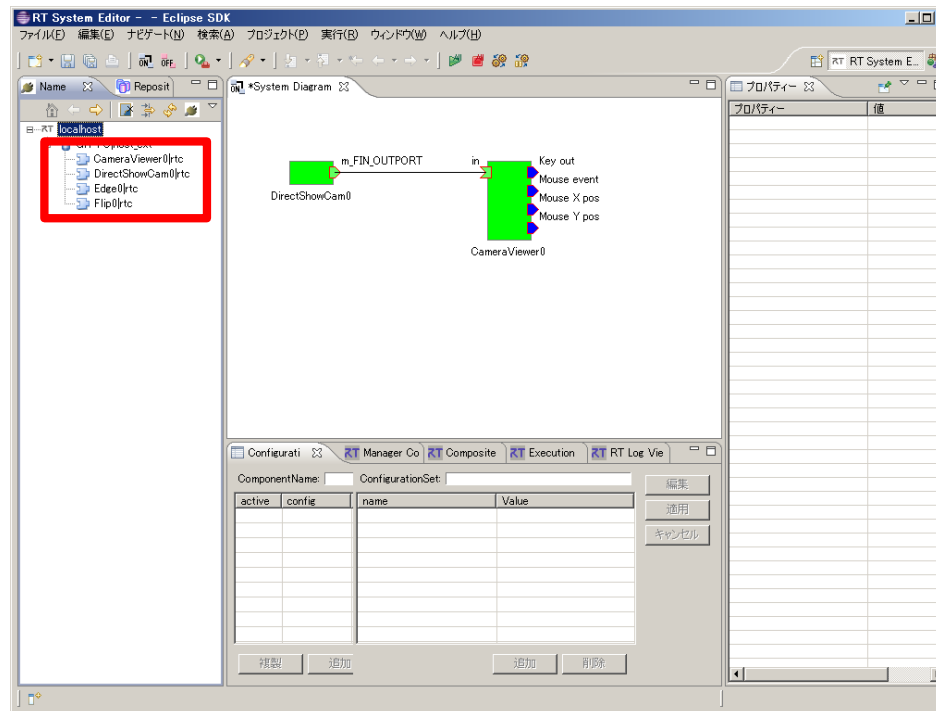
※Buffer Policy = Block+timeout時間の指定で, 一定時間後読み出し/書き込み不可能な場合にタイムアウトを発生させる処理となる

- Flipコンポーネントの起動

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

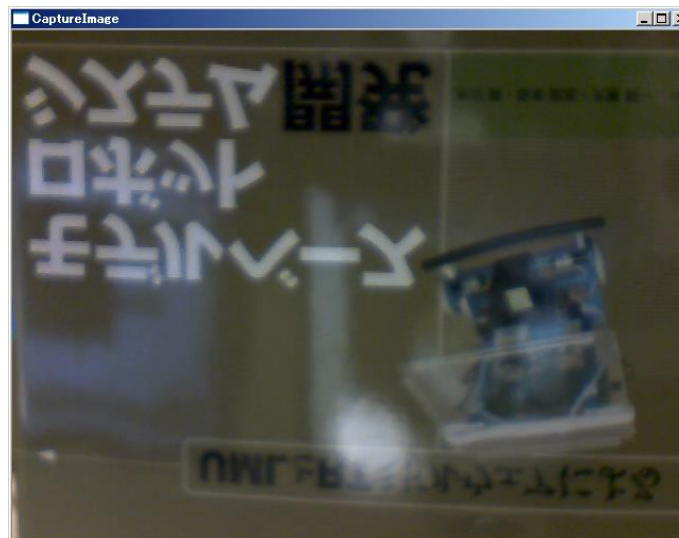
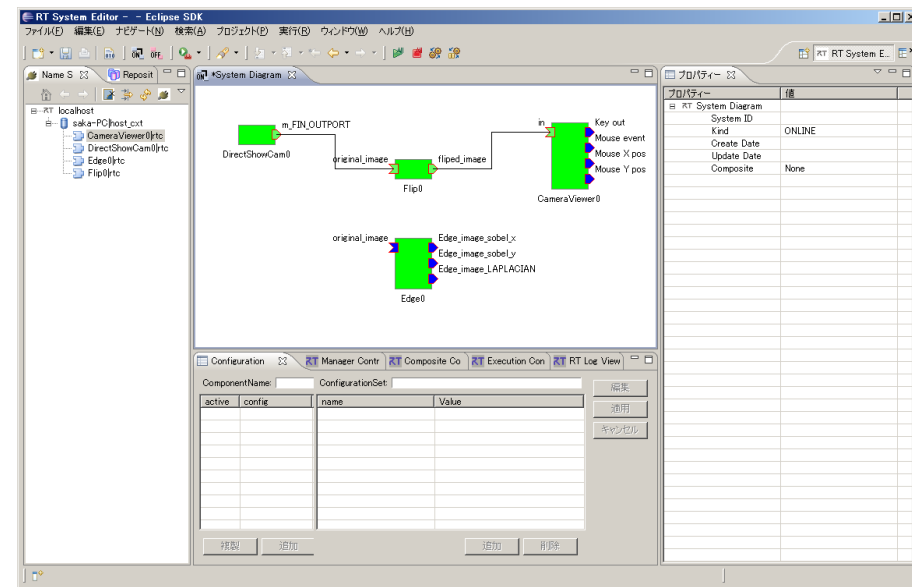
- Edgeコンポーネントの起動

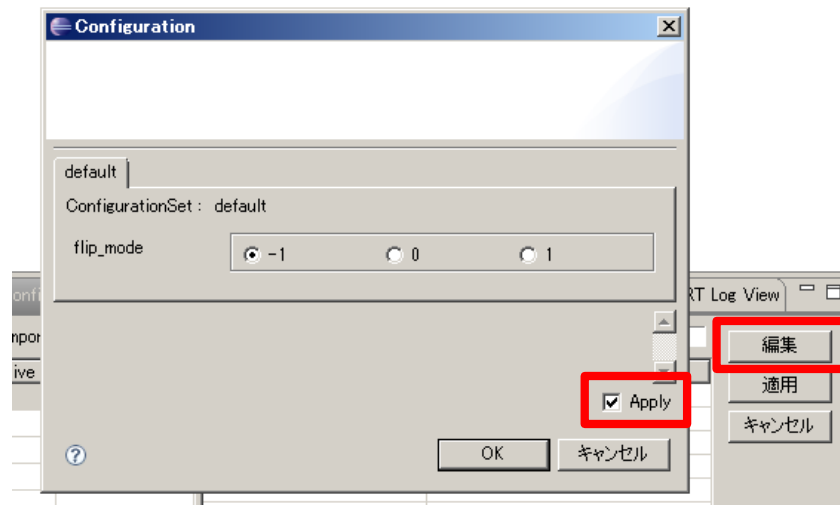
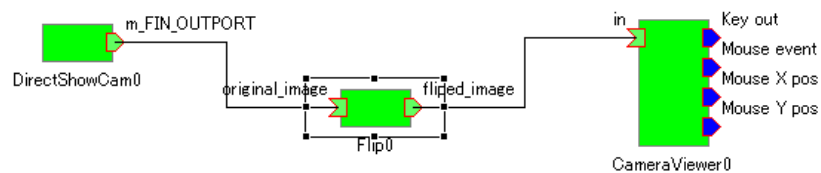
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
→[opencv-rtcs]→ **[EdgeComp.exe]**



Flip側との接続

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

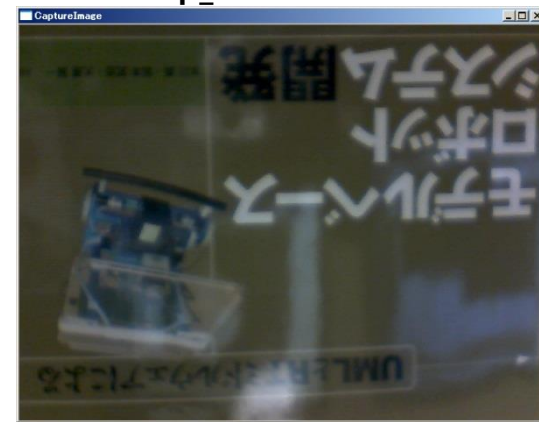
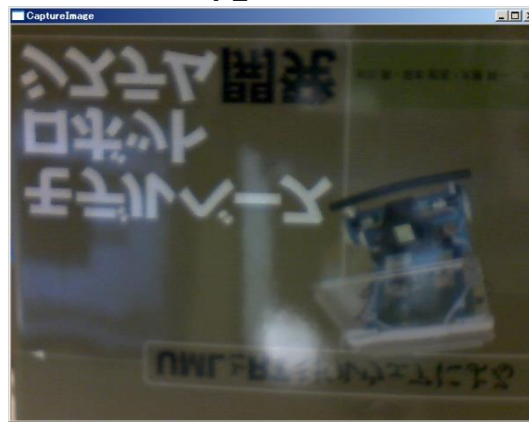




flip_mode=1

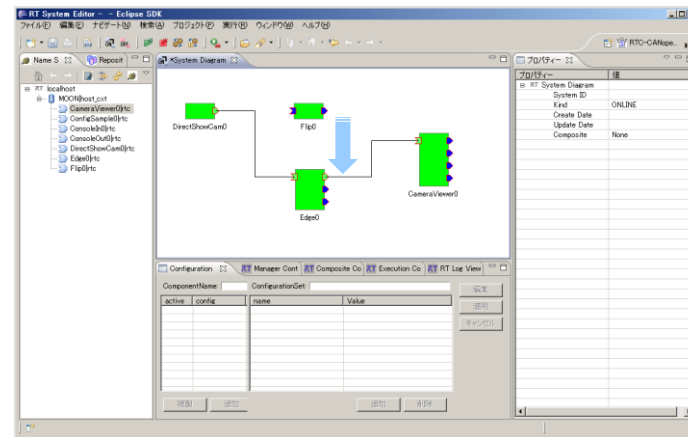
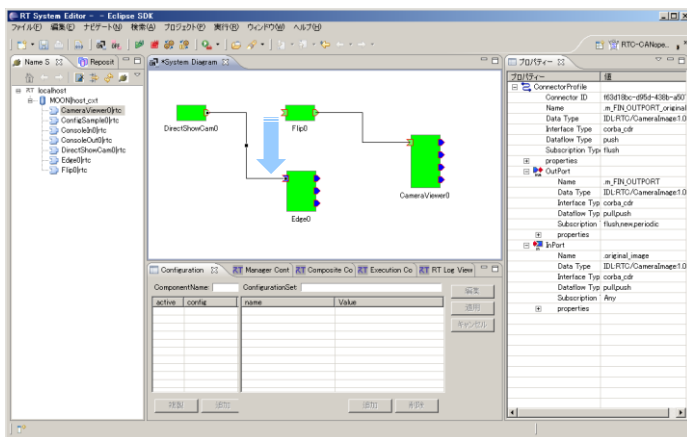
flip_mode=0

flip_mode=-1



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

- Edge側への差し替え
 - Flipに繋がっている接続線を選択
 - Flip側のPort部分に表示されているハンドルをEdge側のPortに繋ぎ替え
 - 接続プロファイルはデフォルト設定のまま



コンポーネント名	処理内容
Affine	入力画像をAffine変換
BackGroundSubtractionSimple	2枚の入力画像の差分抽出
Binarization	入力画像を2値化
Chromakey	入力画像から背景を取り除き, 別の背景に重ね合わせる
DilationErosion	2値化後の画像に膨張縮小処理
Edge	入力画像のエッジ検出
Findcontour	入力画像の輪郭抽出処理
Flip	入力画像の反転処理
Histogram	入力画像内のヒストグラム算出処理
Hough	入力画像をハフ変換
ImageSubstraction	差分画像の抽出
ObjectTracking	指定物体を画像内から抽出
Perspective	遠近透過処理
RockPaperScissors	手の形状認識, グー/チョキ/パーの認識
Rotate	指定角度での画像回転
Scale	入力画像の指定倍への拡大/縮小処理
SepiaComp	入力画像のセピア化処理

- 全て[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]→[opencv-rtcs]内のサンプル

コンポーネント開発ツール RTCBuilderについて

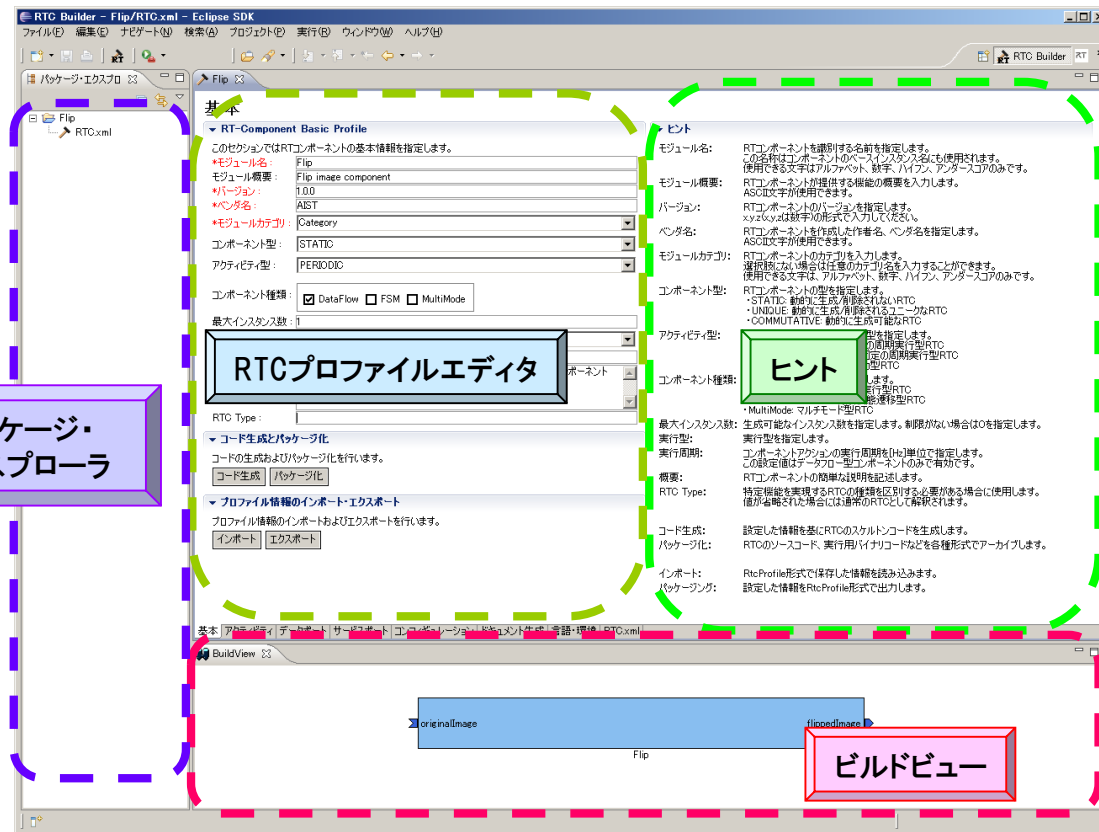
■ RTCBuilderとは？

- 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

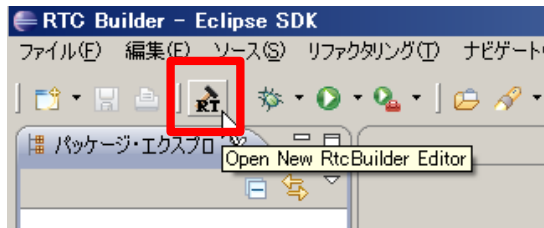
パッケージ・
エクスプローラ

- ※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



ビルドビュー

① ツールバー内のアイコンをクリック



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

※メニューから「ファイル」-「Open New Builder Editor」を選択

※任意の場所にプロジェクトを作成したい場合

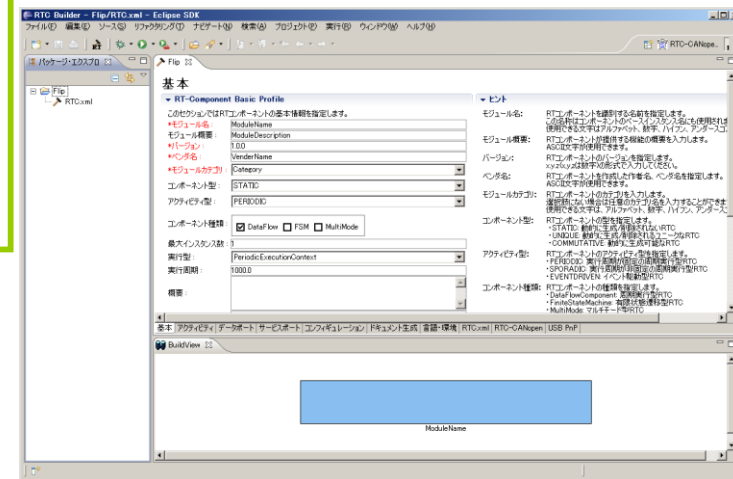
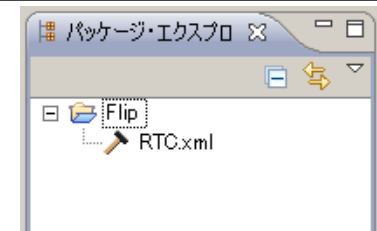
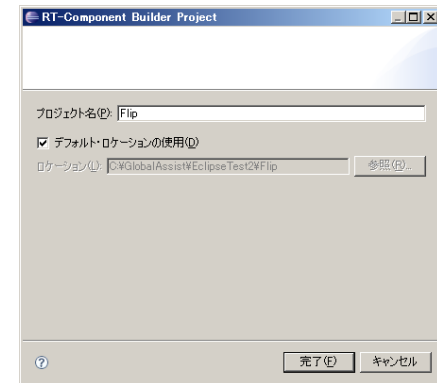
②にて「デフォルト・ロケーションの使用」チェックボックスを外す

「参照」ボタンにて対象ディレクトリを選択

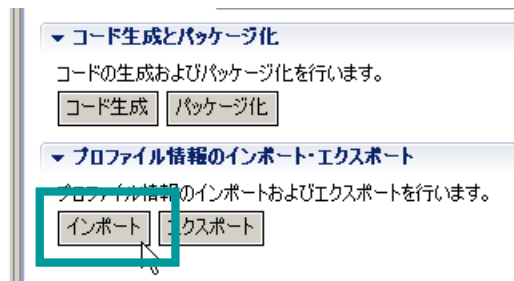
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名: Flip

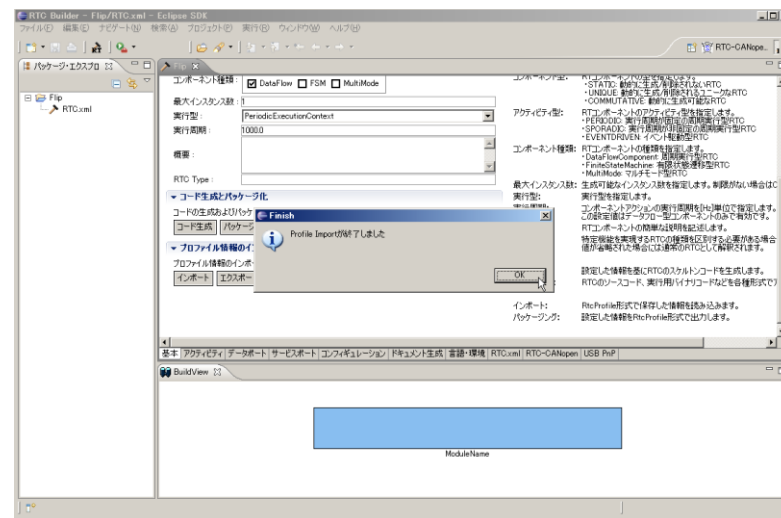
② 「プロジェクト名」欄に入力し、「終了」



①「基本」タブ下部の「インポート」ボタンをクリック



②【インポート】画面にて対象ファイルを選択



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

● コード生成

RTC Type :

▼ **コード生成とパッケージ化**

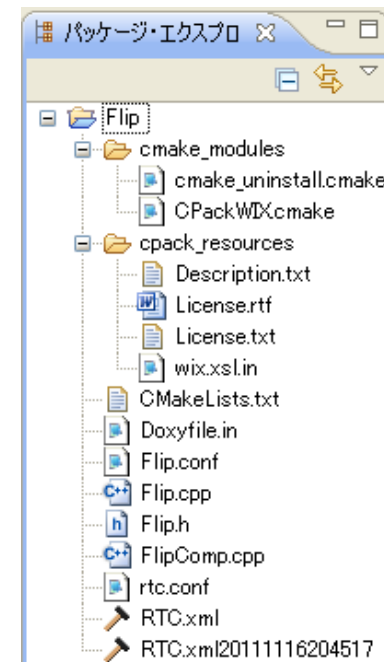
コードの生成およびパッケージ化を行います。

コード生成 **パッケージ化**

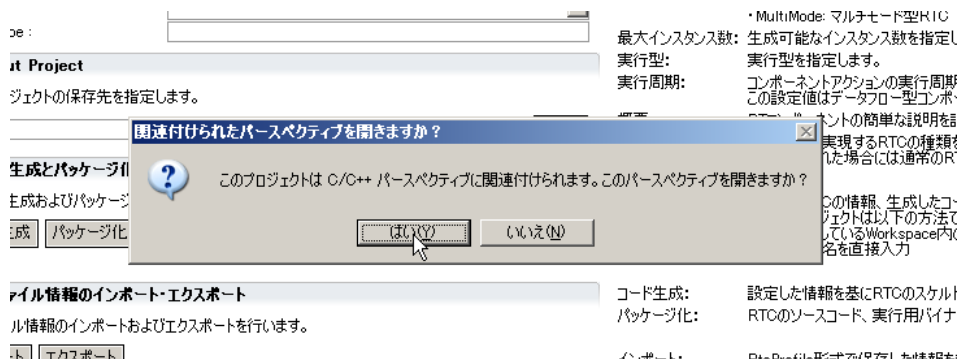
▼ **プロフィール情報のインポート・エクスポート**

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**



● 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

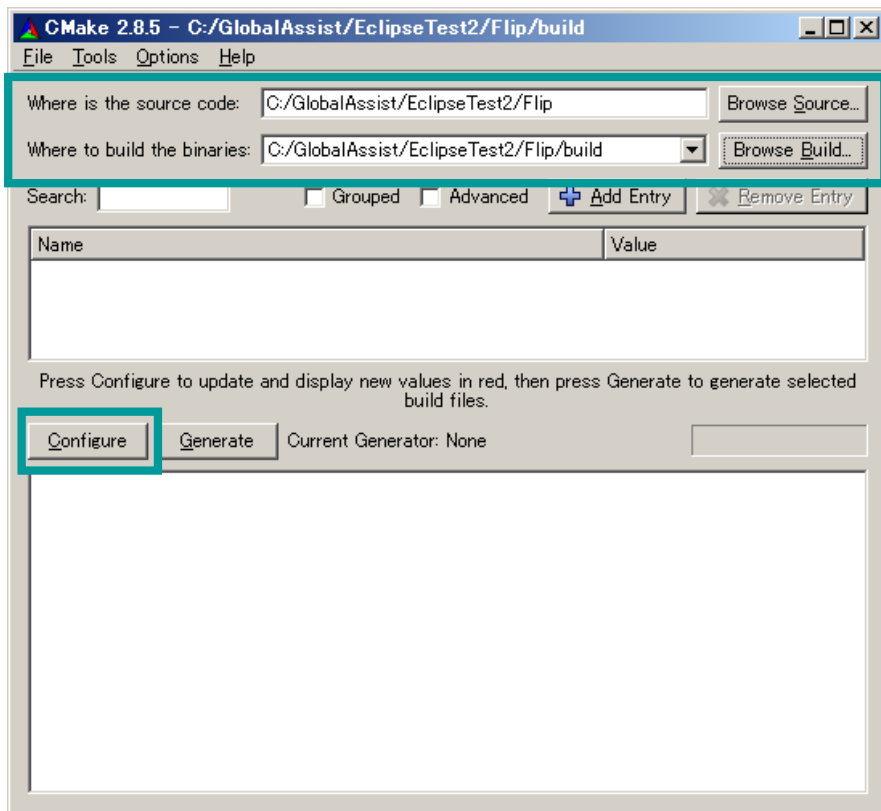
C++版RTC → CDT

Java版RTC → JDT

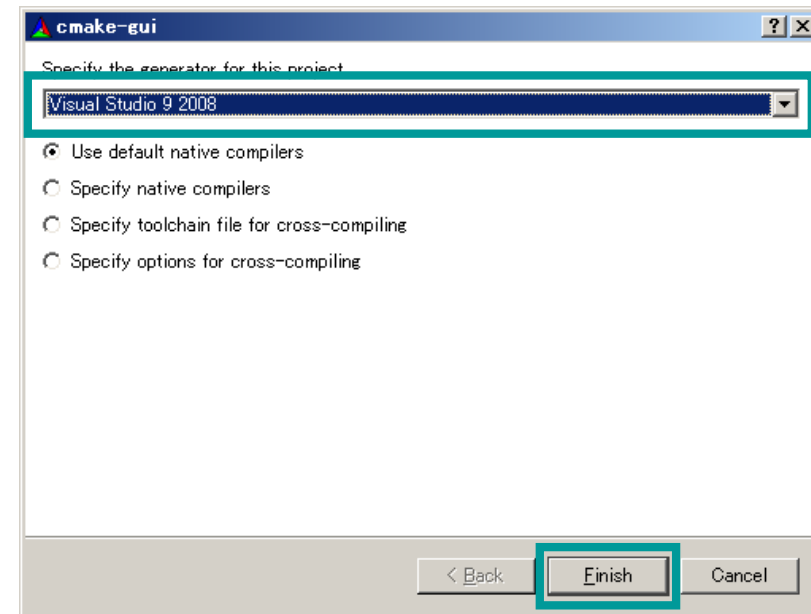
(デフォルトインストール済み)

Python版 → PyDev

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

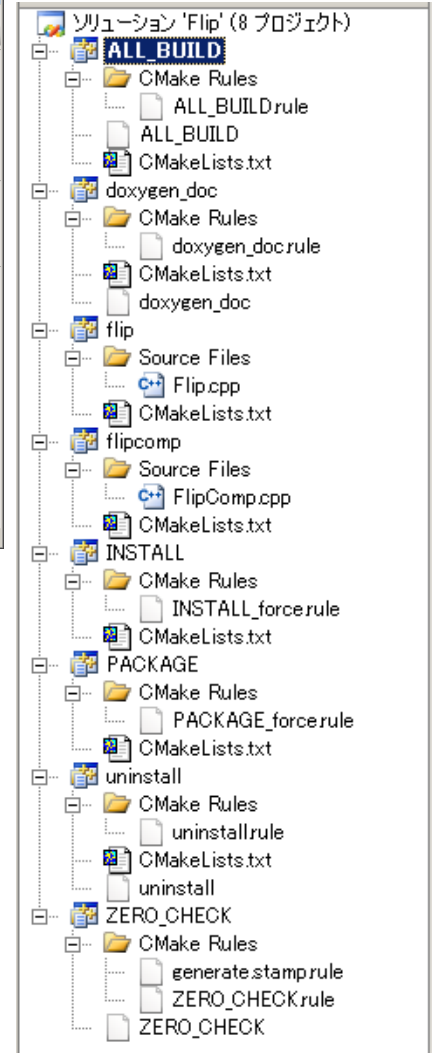
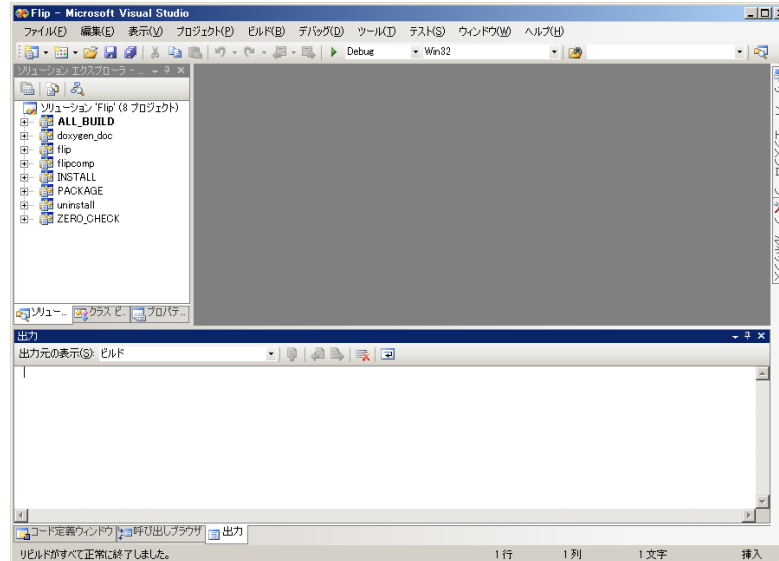
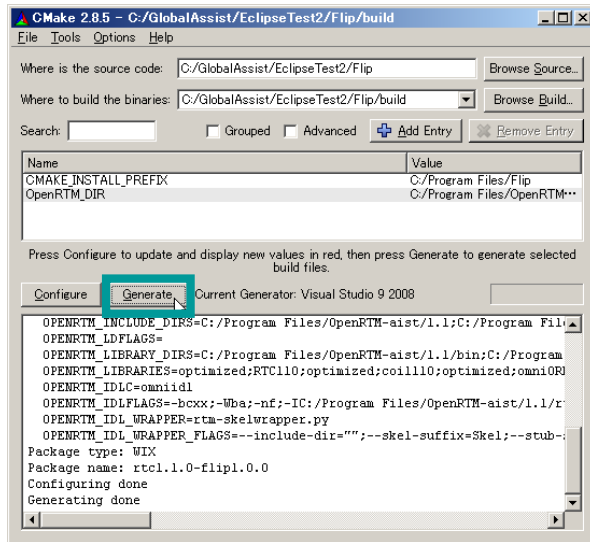


- ② 「Configure」を実行し, 使用するプラットフォームを選択

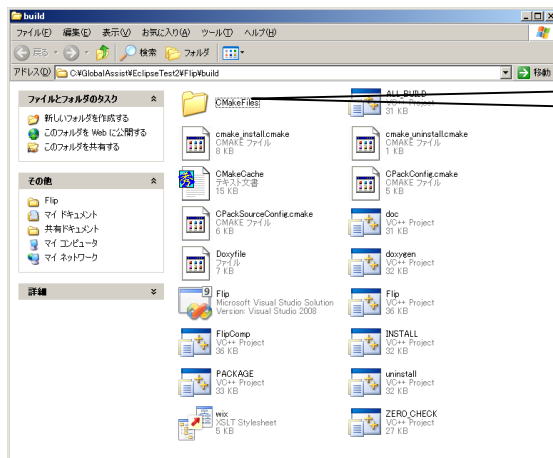


- ※binaryには, sourceとは別のディレクトリを指定する事を推奨
- ※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

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き、「ソリューションをビルド」を実行



Flip
Microsoft Visual Studio Solution
Version: Visual Studio 2008

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

● 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。

▼ 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。

▼ プロファイル情報のインポート・エクスポート

プロファイル情報のインポートおよびエクスポートを行います。

モジュール名: Flip

モジュール概要: 任意(Flip image component)

バージョン: 1.0.0

ベンダ名: 任意(AIST)

モジュールカテゴリ: 任意(Category)

コンポーネント型: STATIC

アクティビティ型: PERIODIC

コンポーネントの種類: DataFlow

最大インスタンス数: 1

実行型: PeriodicExecutionContext

実行周期: 1000.0

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

● 生成対象RTCで実装予定のアクティビティを設定

アクティビティ

▼ アクティビティ

このセクションでは使用するアクションコールバックを指定します。

onInitialize	onFinalize
onStartup	onShutdown
onActivated	onDeactivated
onError	onReset
onExecute	onStateUpdate
onAction	onModeChanged

▼ ヒント

onInitialize: 初期化処理です。コンポーネントライフサイクル開始時に一度だけ呼びされます。常に有効。
onFinalize: 終了処理です。コンポーネントライフサイクルの終了時に一度だけ呼びされます。
onStartup: ExecutionContextが実行を開始するとき一度だけ呼びされます。
onShutdown: ExecutionContextが実行を停止するとき一度だけ呼びされます。
onActivated: 非アクティブ状態からアクティブ化されるとき一度だけ呼びされます。
onDeactivated: アクティブ状態から非アクティブ化されるとき一度だけ呼びされます。
onError: ERROR状態に入る前に一度だけ呼びされます。
onReset: ERROR状態からリセットされ非アクティブ状態に移行するとき一度だけ呼びされます。
onExecute: アクティブ状態時に周期的に呼びされます。
onStateUpdate: onExecuteの後毎回呼びされます。
onRateChanged: ExecutionContextのRateが変更されたとき呼びされます。
onAction: 対応する状態に応じた動作を実行するために呼びされます。
onModeChanged: モードが変更された時に呼びされます。

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

▼ 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

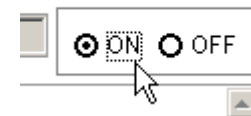
事後条件: コンポーネントの初期化処理が正常に完了している

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

▼ DataPortプロフィール

このセクションではRTCコンポーネントのDataPort(データポート)の情報を設定します。

*ポート名 (InPort)	Add	*ポート名 (OutPort)	Add
originalImage		flippedImage	
	Delete		Delete

▼ Detail

このセクションではデータポート毎の概要を説明するドキュメントを記述します。
上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名: originalImage (InPort)

*データ型: RTC::CameraImage

変数名: originalImage

表示位置: LEFT

Documentation

概要説明: キャプチャされた画像データ

データ型: CameraImage型(OpenRTM-aistのInterfaceDataTypes.idlにて定義されているデータ型)

データ数: 任意

意味: 反転処理の対象となる画像データ

単位: なし

▼ ヒント

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

InPort: RTCコンポーネントにデータを送受信する他のRTCコンポーネントのOutPortと接続する。

OutPort: RTCコンポーネントからデータを送受信する他のRTCコンポーネントのInPortと接続する。

ポート名: データポートを識別するポート名は、同一のコンポーネント内では一意である。ASCII文字列で記述可能。

データ型: データポート間でやり取りするデータの型は、OpenRTMのデータ型を使用する必要があります。

変数名: データポートに関連付けられた変数の名称は任意である。

ポートの場所: RTSystemEditorなどのツールでこのプロファイルはオブジェクトとして使用可能。

ドキュメント: データポートに関する情報を記述する必要がある場合は、このドキュメントに記述する。

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

※ポート)の情報を設定します。

*ポート名 (OutPort)	Add
dp name	
	Delete

② 設定する型情報を一覧から選択

▼ Detail

このセクションではデータポート毎の概要を説明するドキュメントを記述します。
上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名: originalImage (InPort)

*データ型: RTC::CameraImage

変数名: RTC::BumperArrayGeometry

表示位置: RTC::CameraInfo

Documentation

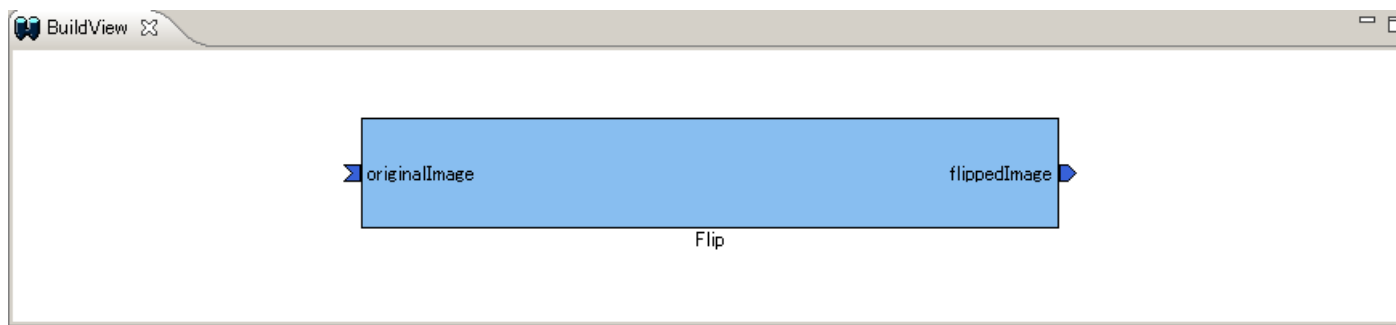
※ データ型は、型定義が記載されたIDLファイルを設定画面にて追加することで追加可能

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

※ 各ポートに対する説明記述を設定可能

→ 記述した各種コメントは、生成コード内にDoxygen形式で追加される

※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::Cameralmage**

変数名: **originalImage**

表示位置: **left**

● OutPort

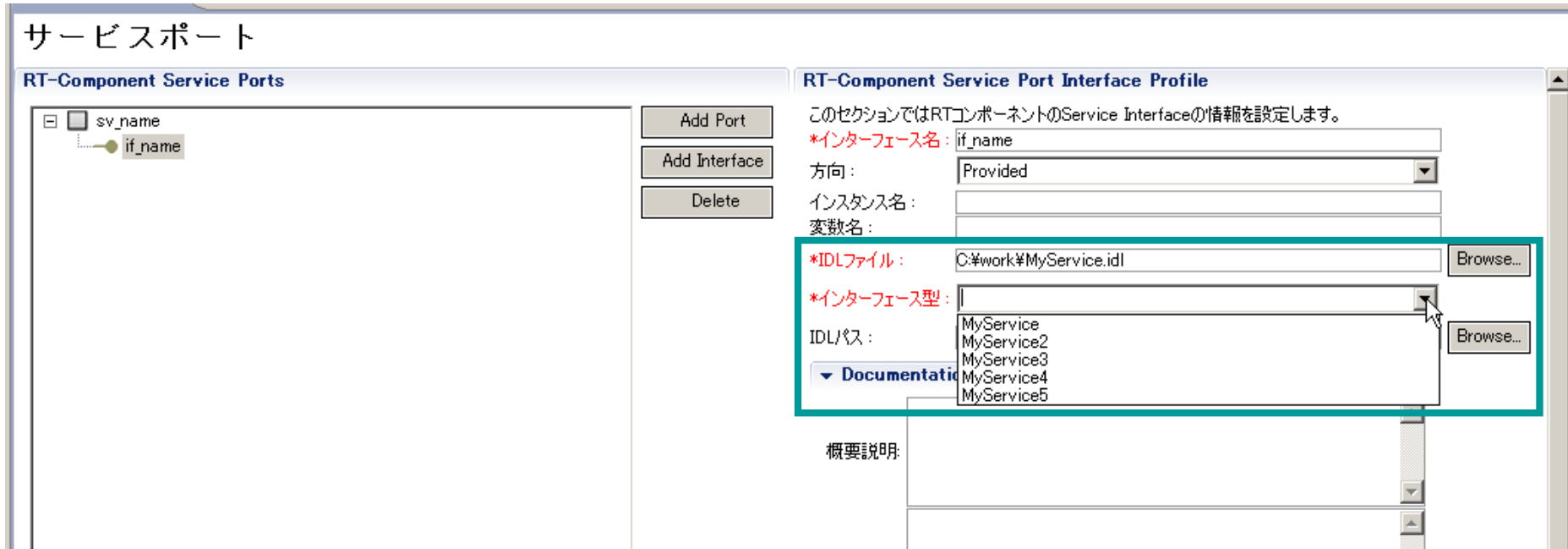
ポート名: **flippedImage**

データ型: **RTC::Cameralmage**

変数名: **flippedImage**

表示位置: **right**

- 生成対象RTCに付加するServicePortの情報を設定



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

今回のサンプルでは未使用

n 生成対象RTCで使用する設定情報を設定

コンフィギュレーション・パラメータ

▼ RT-Component Configuration Parameter Definitions

このセクションではRTコンポーネントのコンフィギュレーション・パラメータを指定します。

*名称	
flipMode	

Add Delete

▼ Detail

このセクションでは各コンフィギュレーション・パラメータの詳細情報を指定します。

パラメータ名: flipMode

*データ型: int

*デフォルト値: 0

変数名: flipMode

単位:

制約条件: [-1,0,1]

Widget: radio

Step:

Documentation

データ名: flipMode

デフォルト値: 0

概要説明: 画像の反転方法を指定するパラメータ

単位: なし

データ範囲: -1,0,1

制約条件: 0: 上下反転したい場合
1: 左右反転したい場合
-1: 上下左右反転したい場合

▼ ヒント

Config. Param: RTコン
コン
再利
パラメ

パラメータ名: コン
パラメ
名前

データ型: コン
基本

デフォルト値: コン
RTコン
解釈

変数名: コン
実際

単位: コン

制約条件: コン
指定
・100
・範囲
・列挙
・配列
・ハッシュ

Widget: コン

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

▼ RT-Component Configuration Parameter Definitions

このセクションではRTコンポーネントのコンフィギュレーション・パラメータを指定します。

*名称	
conf_name0	

Add Delete

②詳細画面にて、型情報、変数名などを設定

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

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

- 制約条件について

- データポートとコンフィギュレーションに設定可能
- チェックはあくまでもコンポーネント開発者側の責務
 - ミドルウェア側で検証を行っているわけではない

- 制約の記述書式

- 指定なし: 空白
- 即値: 値そのもの
 - 例) 100
- 範囲: $<$, $>$, $<=$, $>=$
 - 例) $0 \leq x \leq 100$
- 列挙型: (値1, 値2, ...)
 - 例) (val0, val1, val2)
- 配列型: 値1, 値2, ...
 - 例) val0, val1, val2
- ハッシュ型: { key0: 値0, key1: 値1, ... }
 - 例) { key0: val0, key1: val1 }

- Widget

- text(テキストボックス)
 - デフォルト
- slider(スライダー)
 - 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。

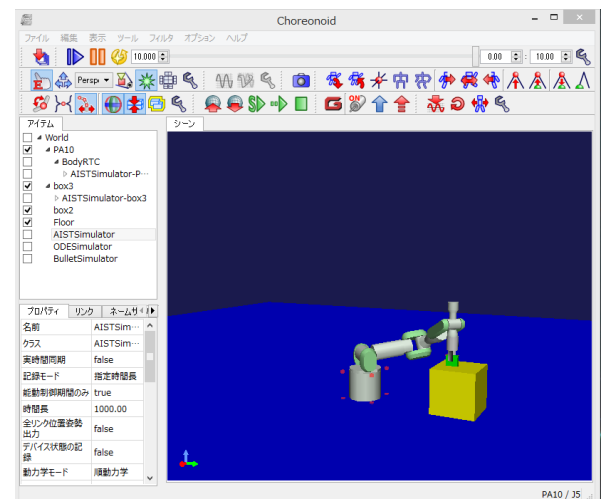
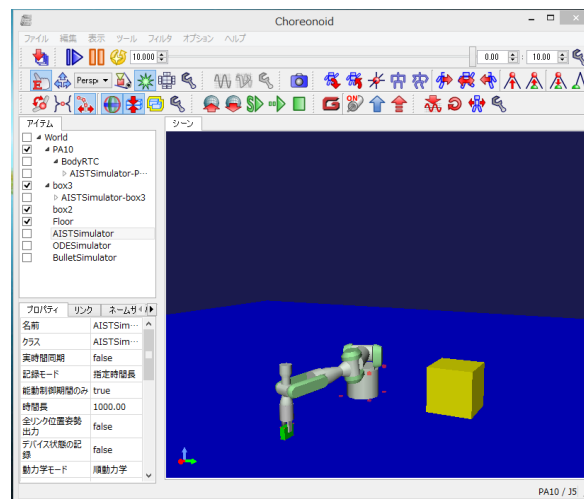
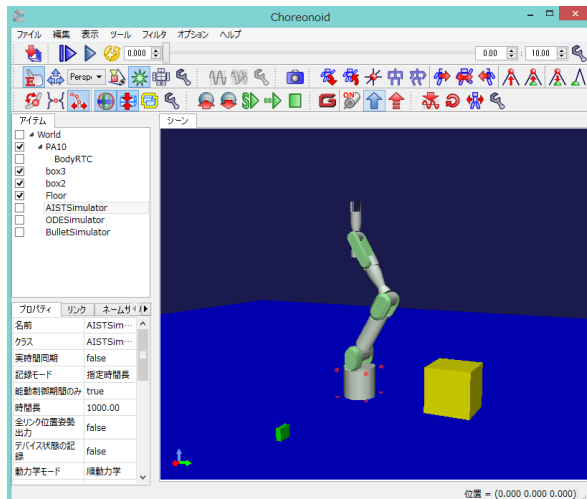
環境: 言語ごとのライブラリの依存関係や、使用するOSなどの環境を選択します。

詳細情報で設定した内容(OS情報、ライブラリ情報など)は、プロファイル内にも保存されます。

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

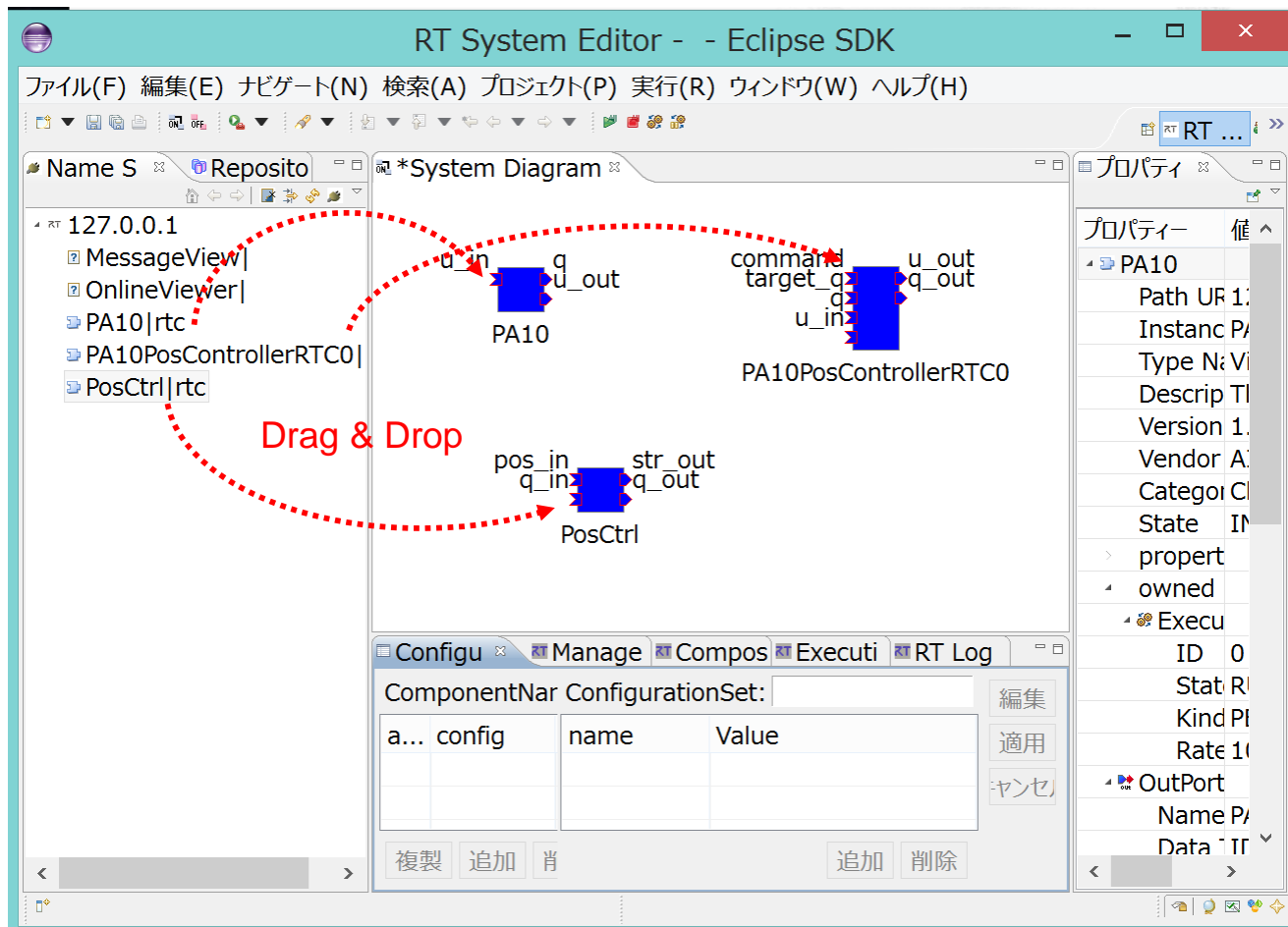
- ロボットシミュレータChoreonoidのPA10を操作するRTCを作成する
- ChoreonoidとPA10のモデル、制御用RTCについては、USBメモリ内のものを利用
- 手先の位置と姿勢を入力してPA10で緑の箱を移動させる



- NameServerの起動: **rtm-naming.bat**
- RT SystemEditorの起動: **OpenRTP.bat**
- Choreonoidの起動: **Choreonoid-PA10.bat**
- PA10のコントローラの起動: **PA10_PosCtrl.bat**

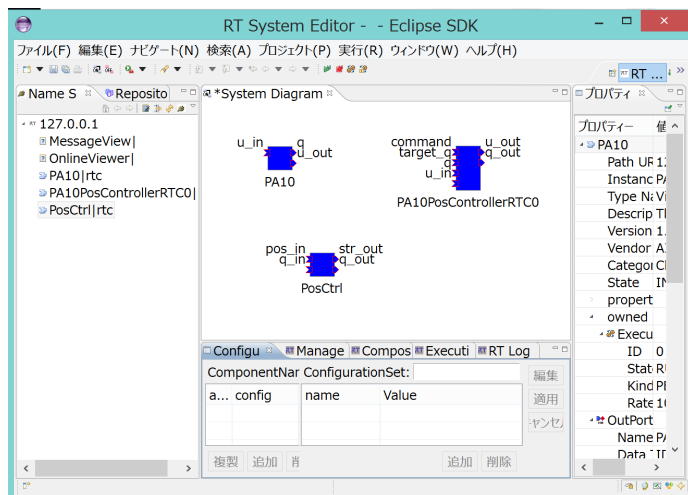
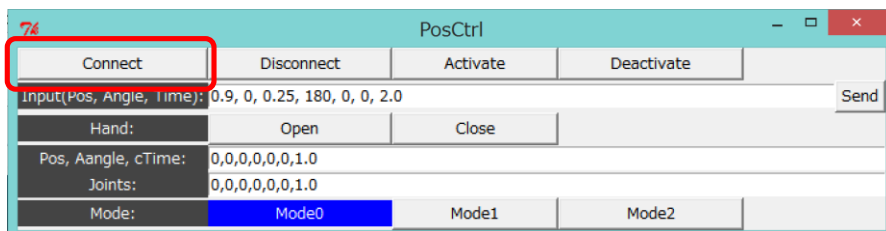
名前	更新日時	種類	サイズ
Choreonoid1.4	2014/07/28 11:11	ファイル フォルダー	
eSEAT	2014/08/04 13:35	ファイル フォルダー	
LeapMotion	2014/07/23 13:56	ファイル フォルダー	
OpenRTM-aist	2014/08/04 15:09	ファイル フォルダー	
Choreonoid-GRobo.bat	2014/08/04 15:00	Windows バッチ フ...	
Choreonoid-PA10.bat	2014/08/04 14:59	Windows バッチ フ...	
GroboDemo.bat	2014/08/05 8:38	Windows バッチ フ...	
LeapMotion.bat	2014/08/05 8:16	Windows バッチ フ...	
LeapRTC.bat	2014/08/05 8:17	Windows バッチ フ...	
OpenRTP.bat	2014/08/05 8:18	Windows バッチ フ...	
PA10_PosCtrl.bat	2014/08/05 8:19	Windows バッチ フ...	
PickAndPlace.txt	2014/07/23 9:41	テキスト ドキュメント	
PosMgr.bat	2014/08/05 8:19	Windows バッチ フ...	
rtc_handle.bat	2014/08/04 14:57	Windows バッチ フ...	
rtm-naming.bat	2014/08/04 15:11	Windows バッチ フ...	

- RT SystemEditorを操作してコンポーネントを表示する

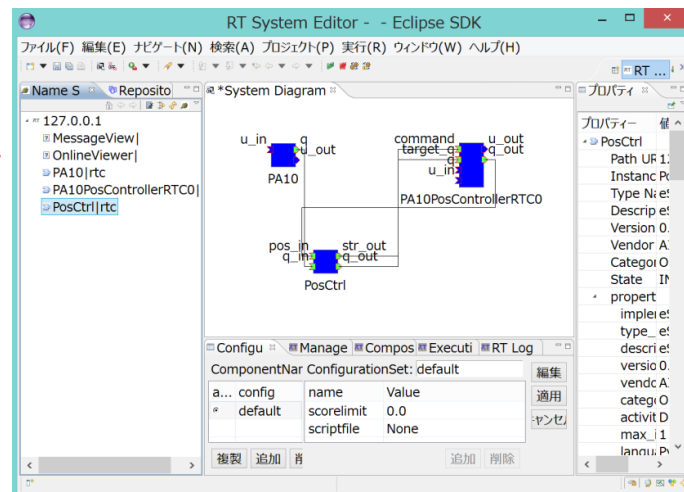


- PA10_PosCtrlの上部のボタンを操作して、コンポーネント間の接続を生成し、RT System Editorで確認する

押下



コネクションの生成



- PA10_PosCtrlの上部のボタンを操作して、コンポーネントを有効化し、Choreonoidのシミュレーションを開始する

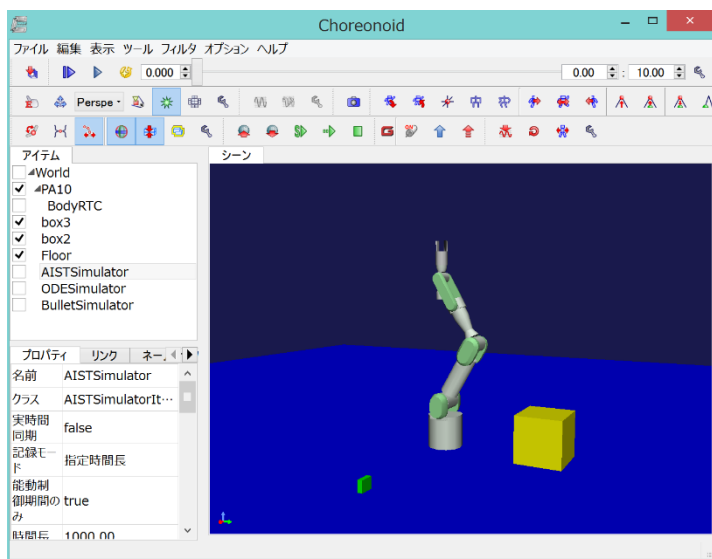
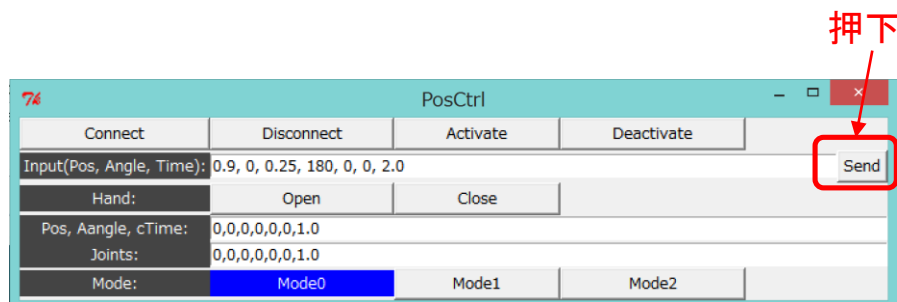
押下

The first screenshot shows the **PosCtrl** window. It has buttons for **Connect**, **Disconnect**, **Activate** (highlighted with a red box), and **Deactivate**. Below these are input fields for **Input(Pos, Angle, Time)** and **Send**. There are also buttons for **Hand: Open** and **Close**, and a **Mode** selector with **Mode0** (selected), **Mode1**, and **Mode2**.

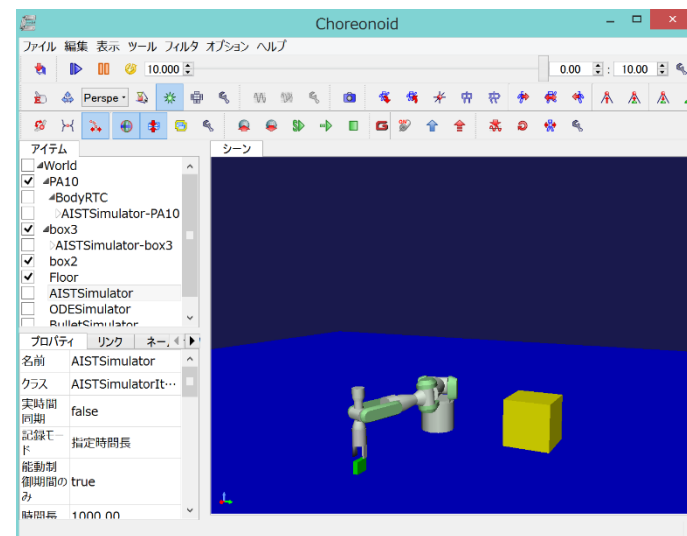
The second screenshot shows the **RT System Editor** in Eclipse SDK. It displays a **System Diagram** with components like **PA10**, **PA10PosControllerRTCO**, and **PosCtrl**. The **Properties** window on the right shows the configuration for **PosCtrl**, including **Path UR 1**, **Instance Name**, **Type Name**, **Description**, **Version**, **Vendor**, **Category**, **State**, **Properties**, **Implementation**, **Type**, **Description**, **Version**, **Vendor**, **Category**, **Activation**, **Max**, and **Language**.

The third screenshot shows the **Choreonoid** window. It has a menu bar with **ファイル**, **編集**, **表示**, **ツール**, **フィルタ**, **オプション**, and **ヘルプ**. Below the menu is a toolbar with various icons. A red box highlights the **Play** button (a green triangle). The main window shows a 3D simulation environment with a robot arm and a yellow cube.

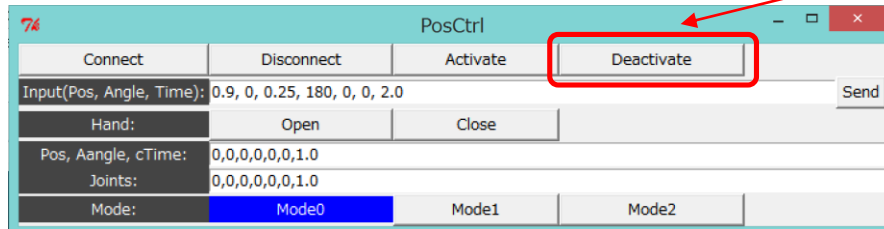
- PA10_PosCtrlを操作して、Choreonoid内のPA10を操作する



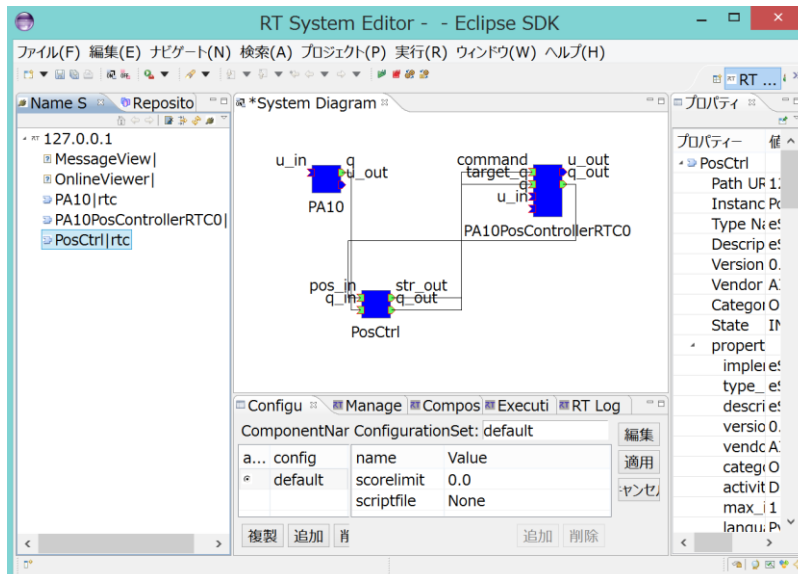
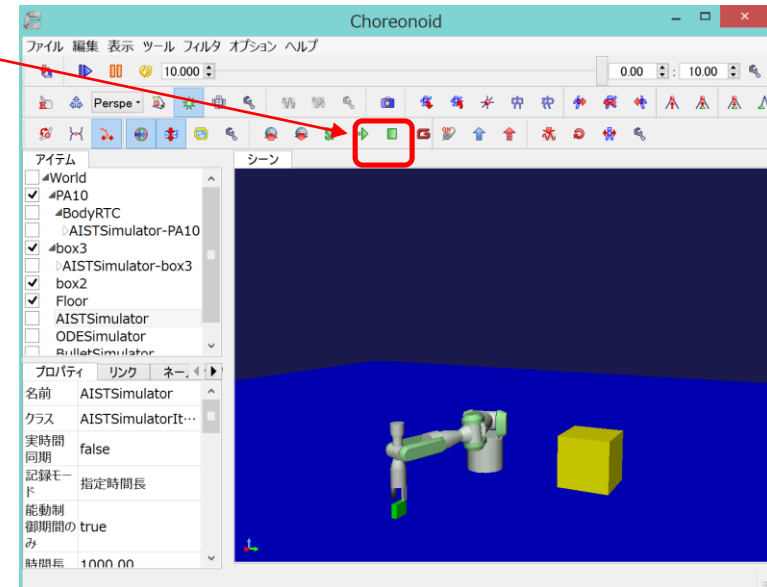
PA10の手先を
目標位置へ移動



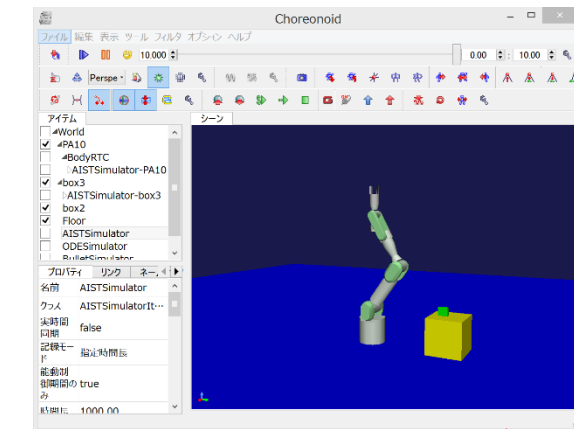
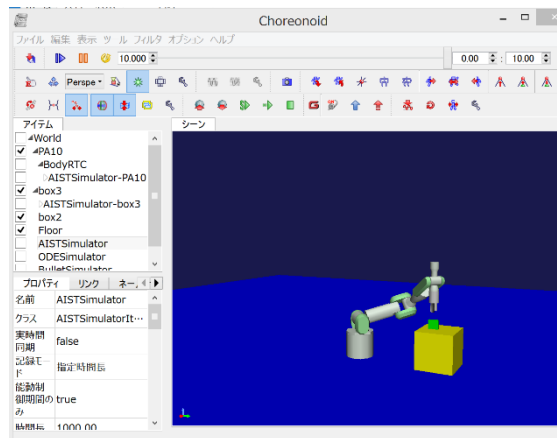
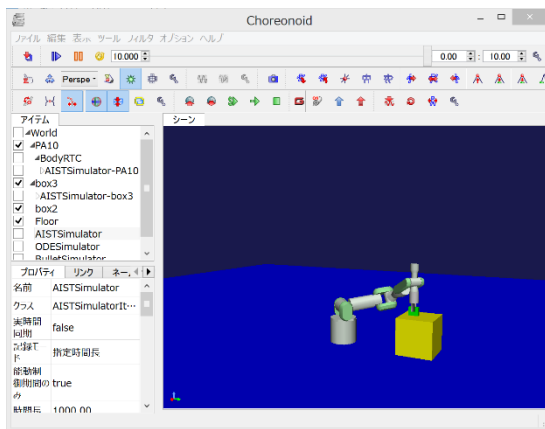
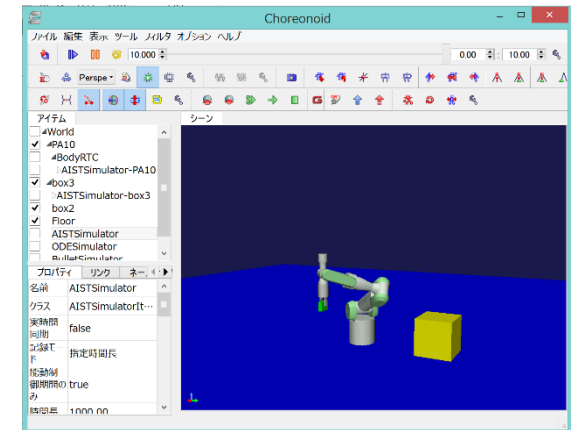
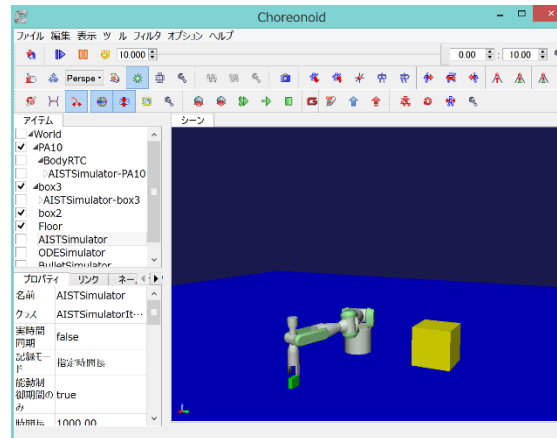
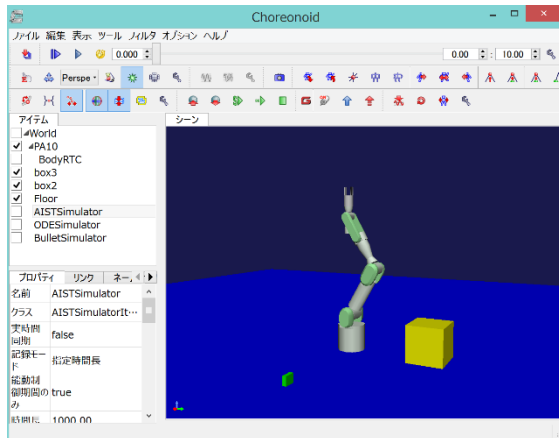
- PA10_PosCtrlの上部のボタンを操作して、コンポーネントを無効化し、Choreonoidのシミュレーションを終了する



押下



- PA10_PosCtrlと同じようにChoreonoid内のPA10を制御して、
緑色の箱を移動させる



- PA10_PosCtrlと同じようにChoreonoid内のPA10を制御して、緑色の箱を移動させるRTCをVC++で実装する。
- 各動作の目標位置は、(X, Y, Z, roll, pitch, yaw)で与える

初期姿勢: 0.0, 0.025, 1.2, 0.0, 0, 0, 2.0

アプローチ点: 0.9, 0.0, 0.25, 180, 0, 0, 2.0

把持点: 0.9, 0.0, 0.20, 180, 0, 0, 2.0

移動中間点: 0.6, 0.0, 0.60, 180, 0, 0, 2.0

移動先アプローチ点: 0.0, 0.65, 0.6, 180, 0, 90, 2.0

移動先: 0.0, 0.65, 0.5, 180, 0, 90, 2.0

退避点: 0.0, 0.65, 0.6, 180, 0, 90, 2.0

最終姿勢: 0.0, 0.025, 1.2, 0.0, 0.0, 0.0, 2.0

- 把持点でハンドを閉じ、移送先でハンドを開けるようにする
- 各移動終了がわかるように、現在のロボットの位置姿勢を読み込むようにする