

第2部

RTコンポーネント作成入門

宮本 信彦

国立研究開発法人産業技術総合研究所
ロボットイノベーション研究センター
ロボットソフトウェアプラットフォーム研究チーム



インストールの確認(Windows)

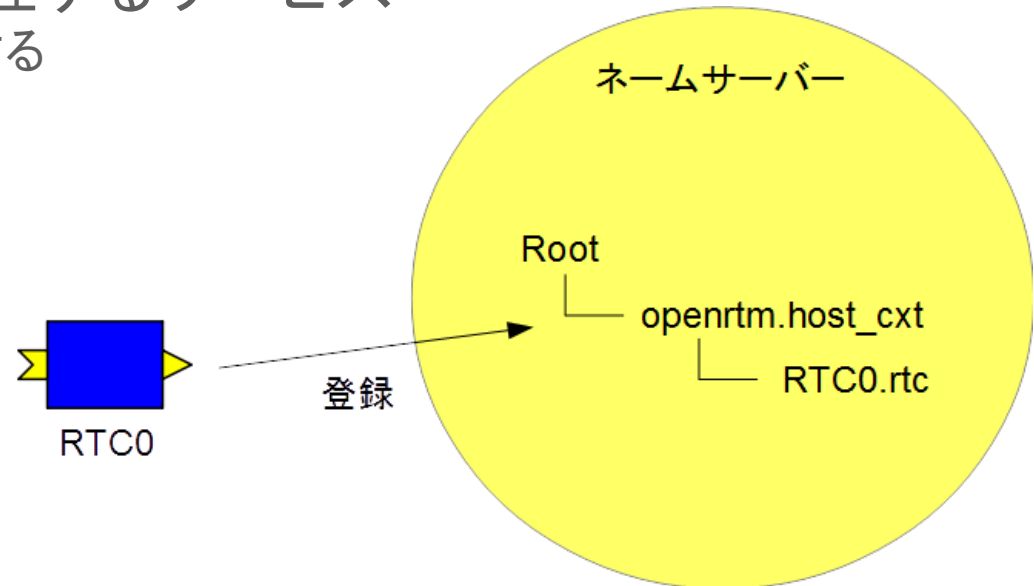
- OpenRTM-aist
 - OpenRTM-aist-1.1.2-RELEASE_x86.msi
 - インストール後に再起動する
 - Visual Studio 2013以外(2010、2012、2015)を使用する場合は環境変数を変更
 - 「RTM_VC_VERSION」をvc10、vc11、vc13
- Python
 - python-2.7.10.msi
 - 2.7.11は府外が発生するため非推奨
 - ※OpenRTM-aistの32bit版をインストールする場合Pythonも32bit版をインストールする。
OpenRTM-aistの64bitをインストールする場合はPythonも64bit版をインストールする。
- PyYAML
 - PyYAML-3.11.win32-py2.7.exe
- CMake
 - cmake-3.5.2-win32-x86.msi
- Doxygen
 - doxygen-1.8.11-setup.exe
- Visual Studio
 - Visual Studio 2013 Community Edition

インストールの確認(Ubuntu)

- OpenRTM-aist
 - `$ sudo sh pkg_install_ubuntu.sh`
- CMake
 - `$ sudo apt-get install cmake cmake-gui`
- Doxygen
 - `$ sudo apt-get install doxygen`
- RT System Editor、RTC Builder
 - eclipse442-openrtp112v20160526-ja-linux-gtk-x86_64.tar.gzを適当な場所に展開
- Java
 - `$ sudo apt-get default-jre`
- OpenCV
 - `$ sudo apt-get install libopencv-dev libcv2.4 libcvaux2.4 libhighgui2.4`
- OpenCVのサンプルコンポーネント
 - 自分でビルドする
 - `$ svn co http://svn.openrtm.org/ImageProcessing/trunk/ImageProcessing/opencv/`
 - `$ cd opencv`
 - `$ mkdir work`
 - `$ cd work`
 - `$ cmake ..`
 - `$ make`
 - `$ sudo make install`
- Code::Blocks(任意)
 - `$ sudo apt-get install codeblocks`

ネームサーバーの起動

- オブジェクトを名前で管理するサービス
 - RTCを一意の名前で登録する



- 起動する手順
 - Windows 7
 - 「スタート」→「すべてのプログラム」→「OpenRTM-aist 1.1.2」→「Tools」→「Start Naming Service」
 - Windows 8.1
 - 「スタート」→「アプリレビュー(右下矢印)」→「OpenRTM-aist 1.1.2」→「Start Naming Service」
 - Ubuntu
 - \$ rtm-naming

ネームサーバーの起動

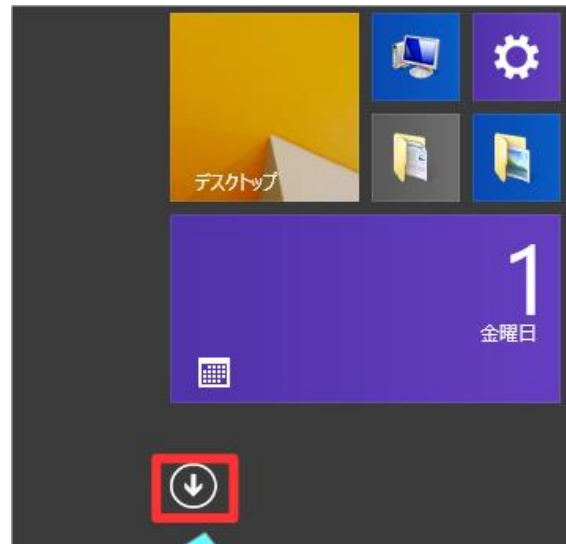
- Windows 8.1

デスクトップ



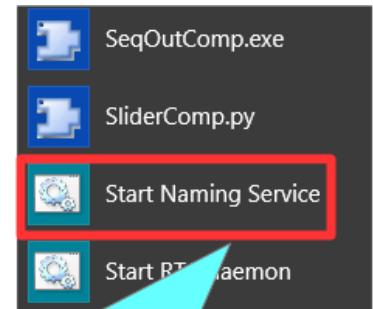
画面左下のアイコンをクリック

スタート画面



画面左下の矢印をクリック

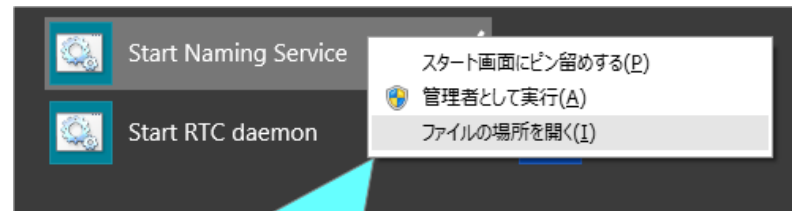
アプリビュー



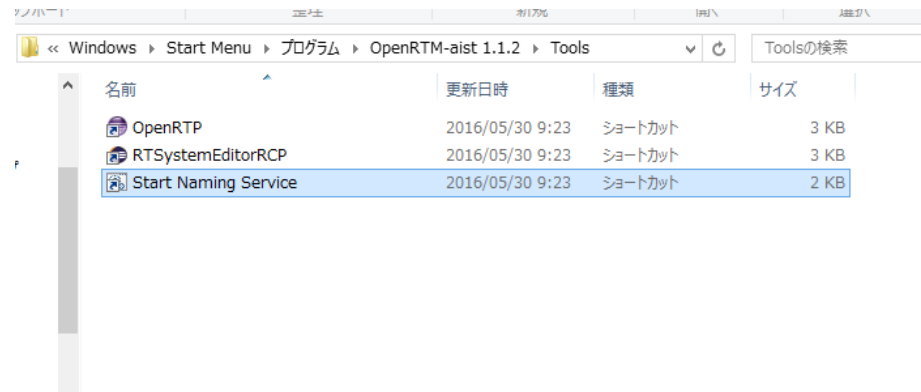
Start Naming Serviceをクリック

ネームサーバーの起動

- いちいちアプリビューから起動するのは非常に手間がかかるため、以下の作業をしてスタートメニューのフォルダを開いておくことをお勧めします。



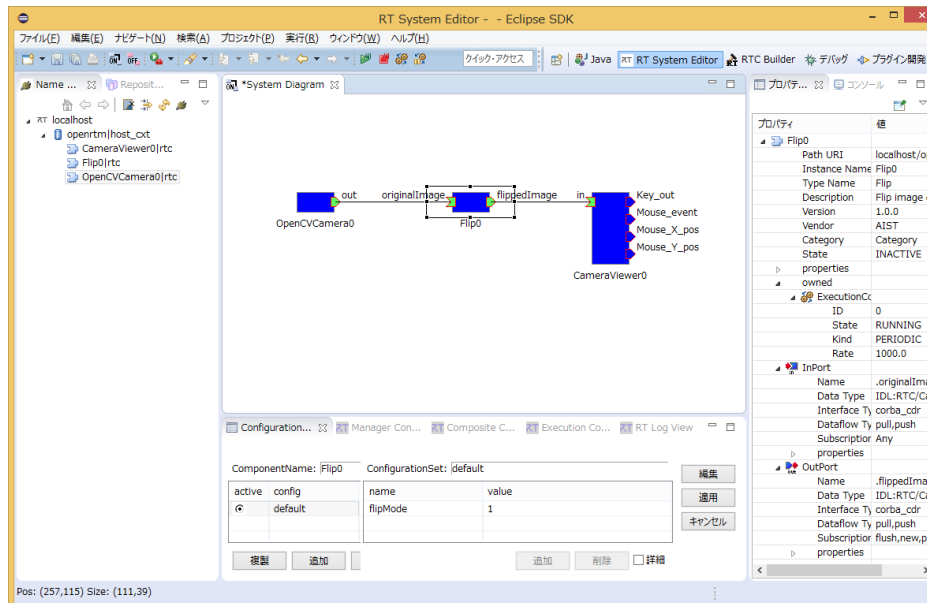
Start naming Serviceを右クリックして
「ファイルの場所を開く」を選択



システム構築支援ツール RT System Editorについて

RT System Editor

- RTCをGUIで操作するためのツール
 - データポート、サービスポートの接続
 - アクティブ化、非アクティブ化、リセット、終了
 - コンフィギュレーションパラメータの操作
 - 実行コンテキストの操作
 - 実行周期変更
 - 実行コンテキストの関連付け
 - 複合化
 - マネージャからRTCを起動
 - 作成したRTシステムの保存、復元



RT System Editorの起動

- 起動する手順
 - Windows 7
 - 「スタート」→「すべてのプログラム」→「OpenRTM-aist 1.1.2」→「Tools」→「OpenRTP」
 - Windows 8.1
 - 「スタート」→「アプリビュー(右下矢印)」→「OpenRTM-aist 1.1.2」→「OpenRTP」
 - ※先ほどの作業でスタートメニューのフォルダを開いている場合はOpenRTPのショートカットをダブルクリックしても開けます
 - ※同じフォルダに「RTSystemEditorRCP」がありますが、これはRTC Builderが使えないので今回は「OpenRTP」を起動してください。
 - Ubuntu
 - Eclipseを展開したディレクトリに移動して以下のコマンド
 - \$./openrtp

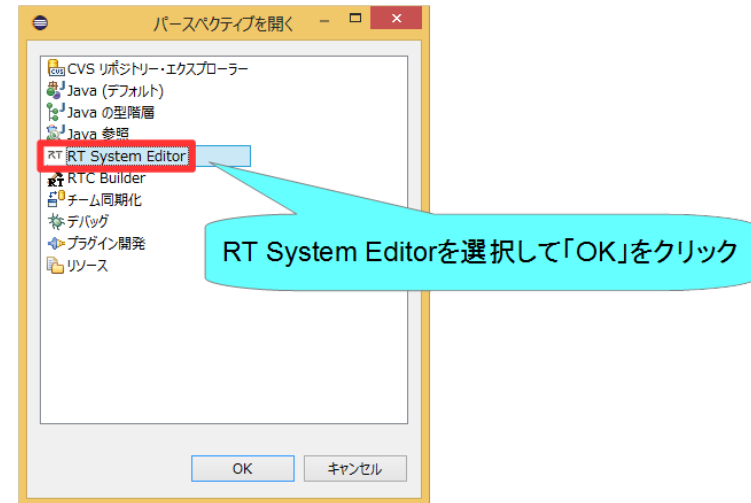
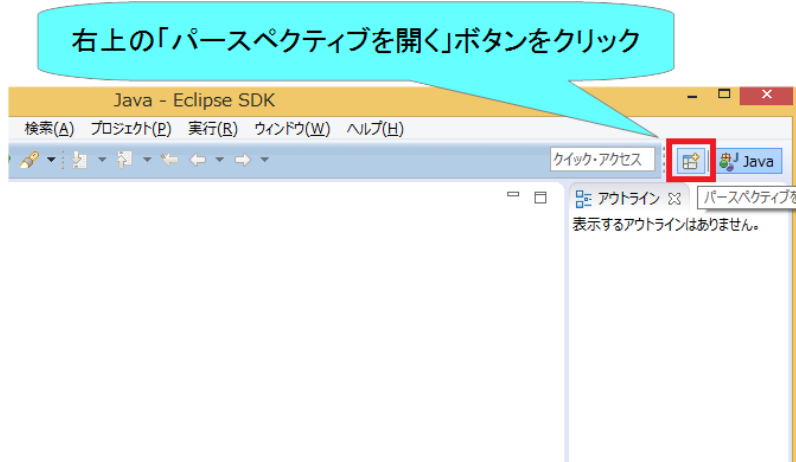
RT System Editorの起動



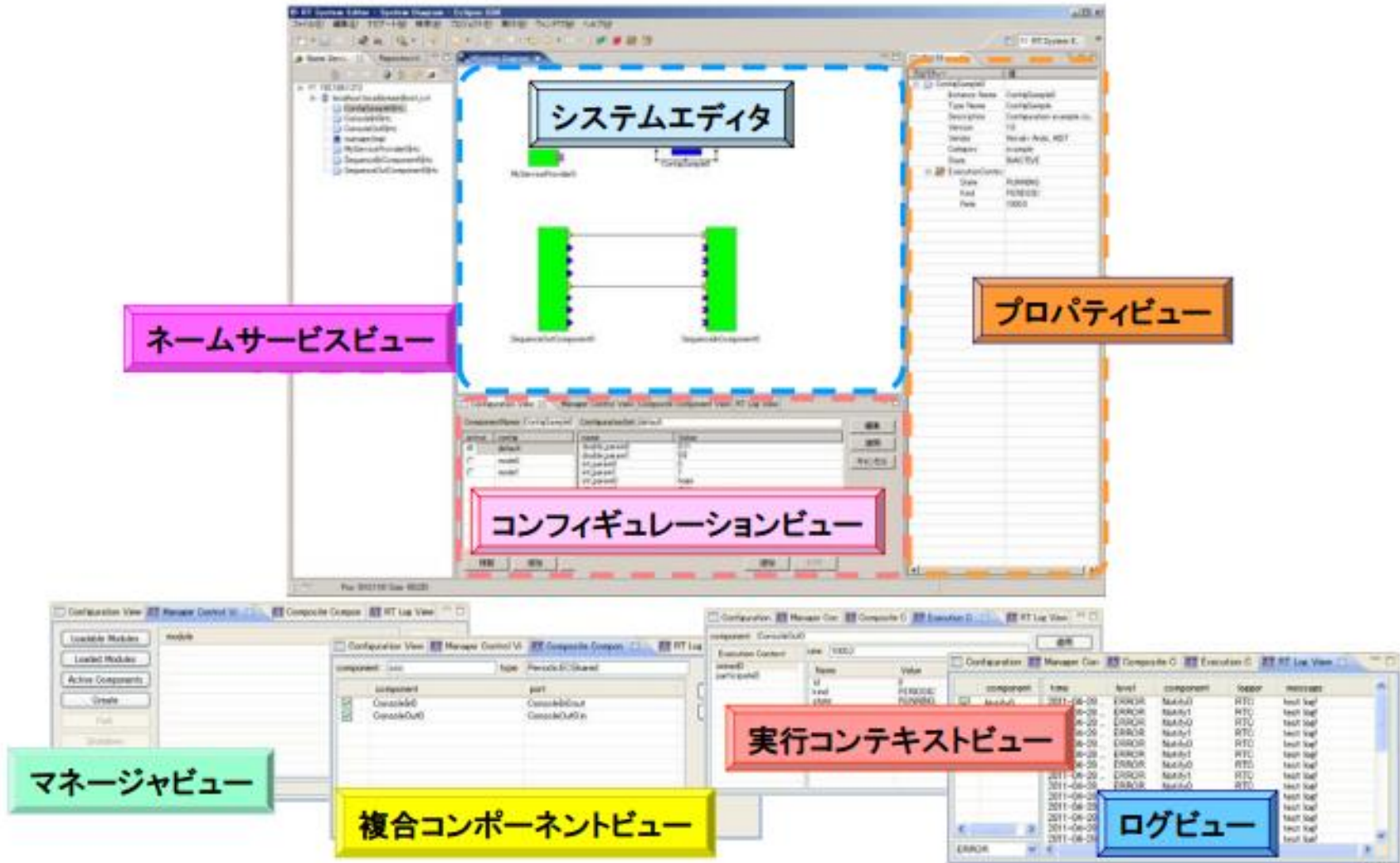
最初に起動したときはWelcomeページが開くので×を押して閉じる



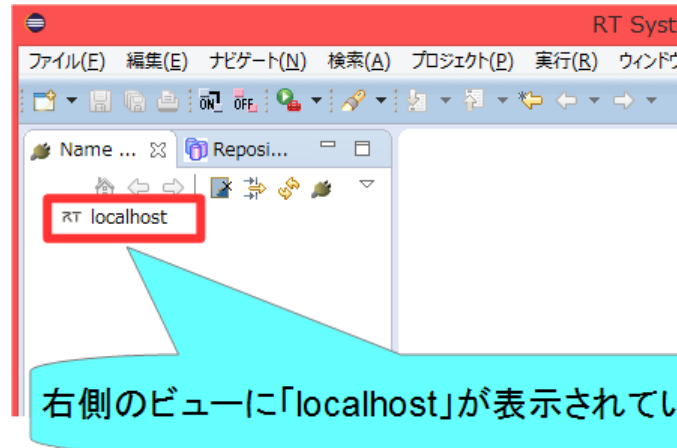
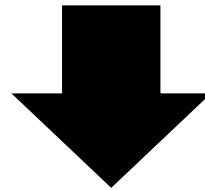
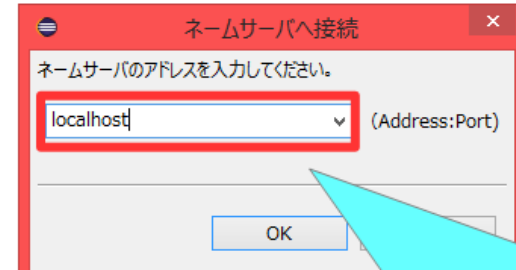
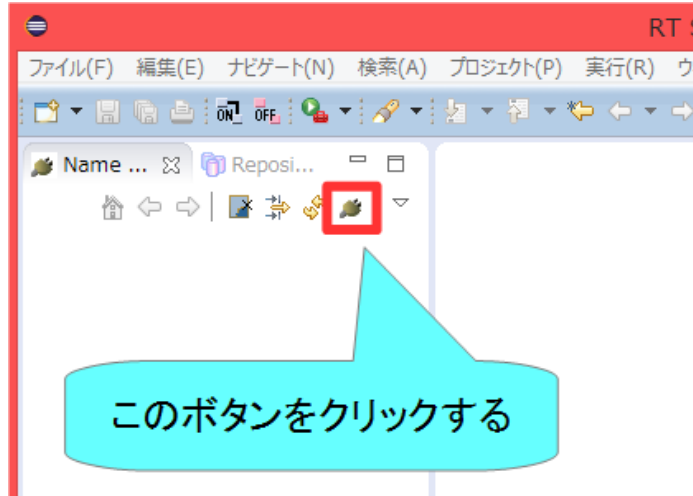
RT System Editorの起動



RT System Editorの画面構成

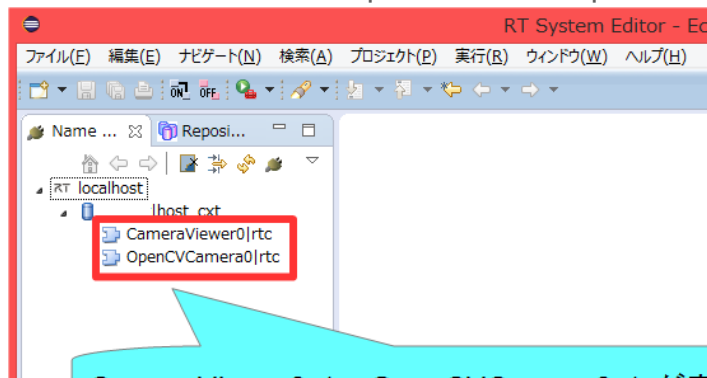


ネームサーバーへ接続



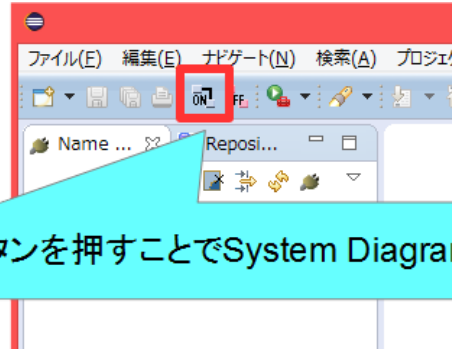
動作確認

- サンプルのRTCを起動して簡単な動作確認を行う
 - ポートの接続
 - アクティブ化
- サンプルのRTCの起動手順
 - CameraViewerコンポーネント、OpenCVCameraコンポーネントを起動
 - Windows 7
 - 「スタート」→「すべてのプログラム」→「OpenRTM-aist 1.1.2」→「Components」→「OpenCV-Examples」→「OpenCVCameraComp.exe」、「CameraViewerComp.exe」
 - Windows 8.1
 - 「スタート」→「アプリビュー(右下矢印)」→「OpenRTM-aist 1.1.2」→「OpenCVCameraComp.exe」、「CameraViewerComp.exe」
 - Ubuntu
 - `$ /usr/share/openrtm-1.1/components/c++/opencv-rtcs/CameraViewerComp`
 - `$ /usr/share/openrtm-1.1/components/c++/opencv-rtcs/OpenCVCameraComp`

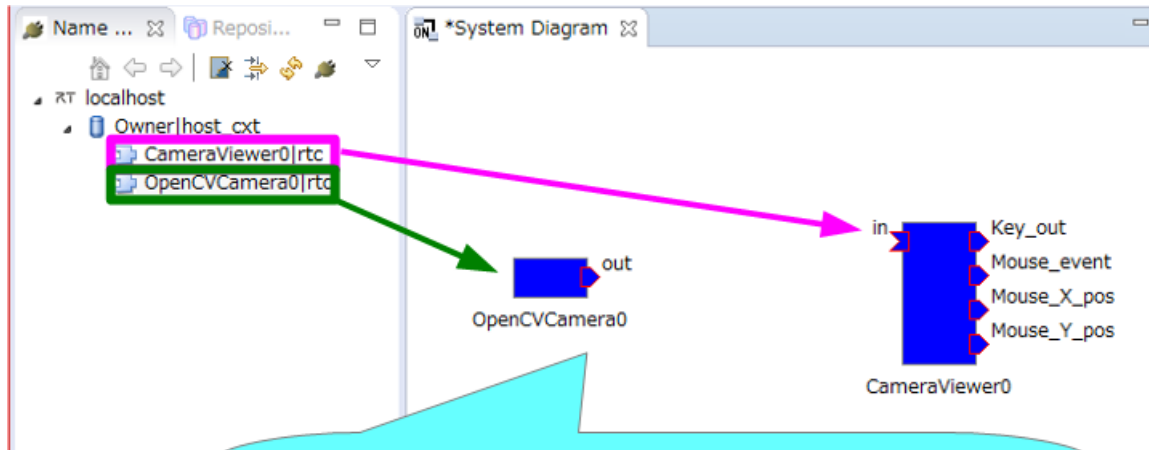


CameraViewer0.rtc、OpenCVCamera0.rtcが表示される

データポートの接続



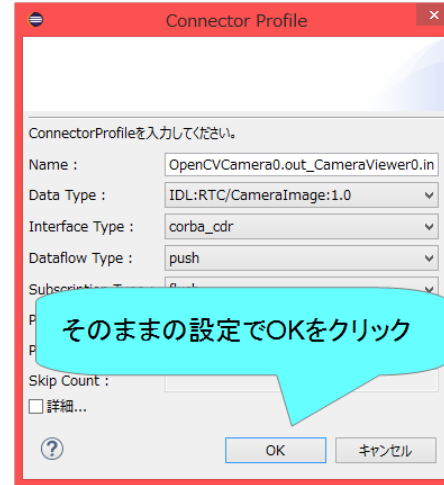
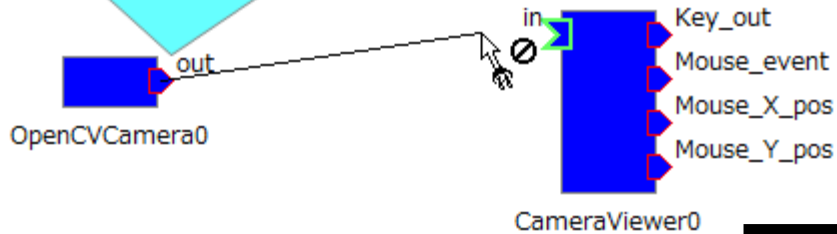
このボタンを押すことでSystem Diagramを表示する



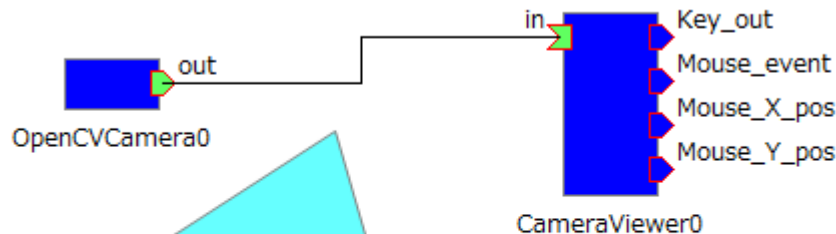
左側のネームサービスビューから
CameraViewer0.rtc、OpenCVCamera0.rtcを
System Diagramにドラッグアンドドロップ

データポートの接続

OpenCVCamera0の「out」を選択して、
CameraViewer0の「in」にドラッグアンドドロップ

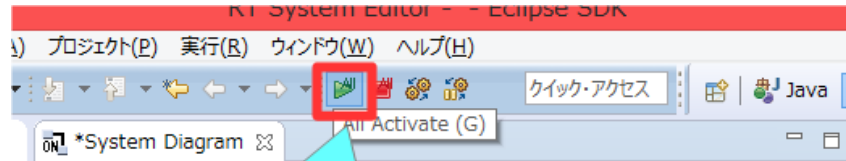


※コネクタプロファイルについては後で説明します

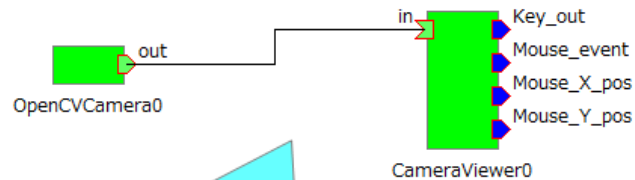
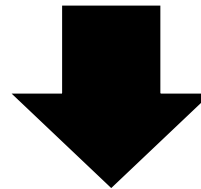


1. ポートの上に線が表示される
2. InPort、OutPortが緑色で表示される

アクティブ化



「All Activate」ボタンを押す

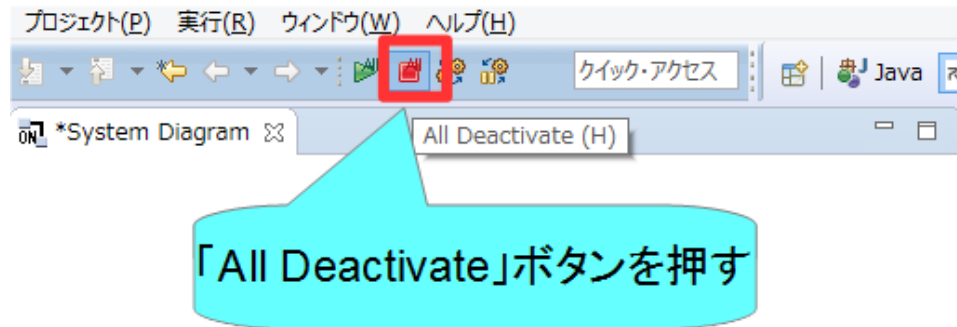


RTCが緑色になればアクティブ化成功

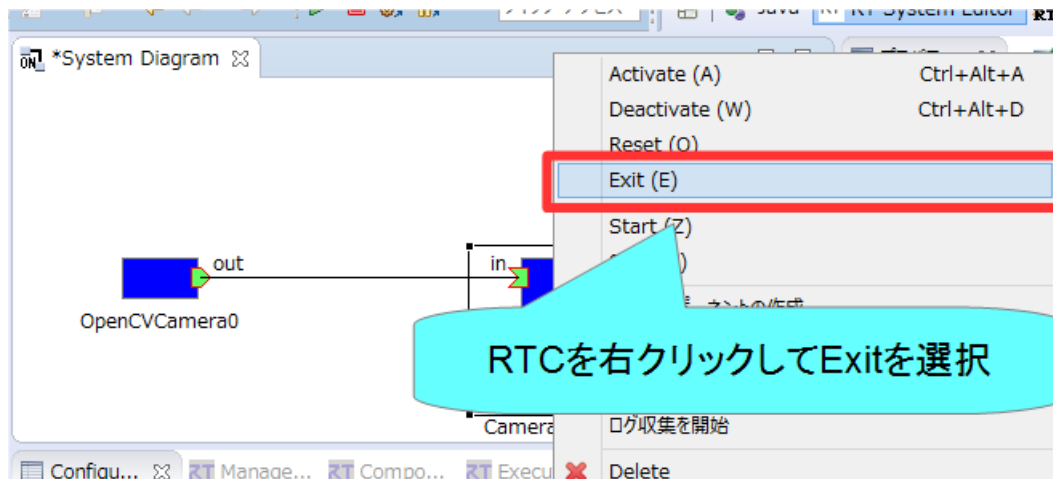
- カメラでキャプチャした画像が表示されるかを確認してください
 - 表示されない場合
 - カメラがPCに接続されていない
 - データポートを接続していない
 - RTCがアクティブになっていない

非アクティブ化、終了

- 非アクティブ化



- 終了

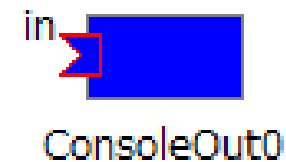
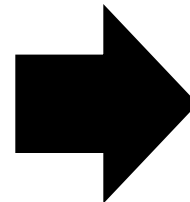


リセット

- RTCがエラー状態に遷移した場合にエディタ上には赤く表示される。



- 以下の操作で非アクティブ状態に戻す



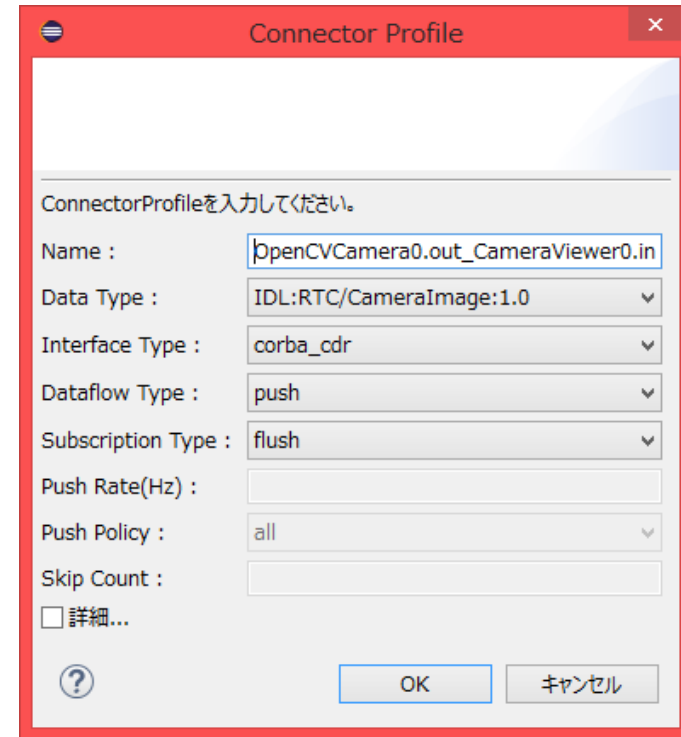
RT System Editor 補足

コネクタプロファイルの設定

項目	設定内容
Name	接続の名称
Data Type	ポート間で送受信するデータの型. ex)TimedOctet, TimedShortなど
Interface Type	データを送信方法. ex)corba_cdrなど
Data Flow Type	データの送信手順. ex)push, pullなど
Subscription Type	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). Subscription TypeがPeriodicの場合のみ有効
Push Policy	データ送信ポリシー. Subscription Type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

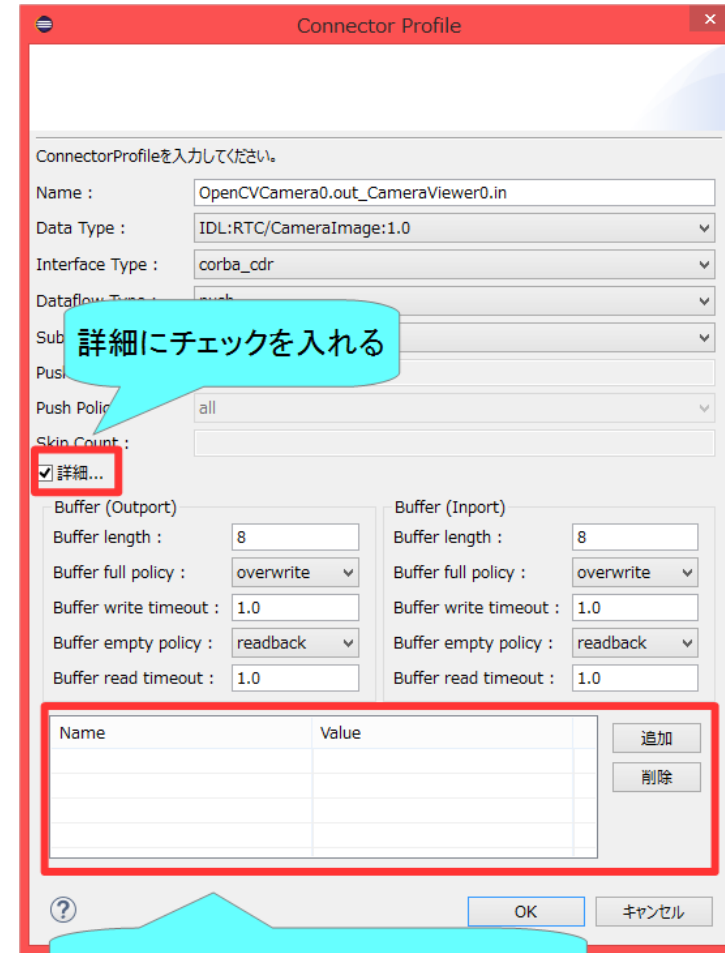
コネクタプロファイルの設定

- InterfaceType
 - データの送信方法
 - 1.1.2ではcorba_cdr(CORBAによる通信)のみ選択可能
 - 1.2.0では以下の通信方法も選択可能になる予定
 - direct(同一プロセスで起動したRTC間でデータを直接変数に渡す)
 - shared_memory(共有メモリによる通信)
- DataFlowType
 - データの送信手順
 - Push
 - OutPortがInPortにデータを送る
 - Pull
 - InPortがOutPortに問い合わせでデータを受け取る
- SubscriptionType
 - データ送信タイミング(DataFlowTypeがPush型のみ有効)
 - flush(同期)
 - バッファを介さず即座に同期的に送信
 - new(非同期)
 - 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(非同期)
 - 一定周期で定期的にデータを送信
- Push Policy(SubscriptionTypeがnew、periodicのみ有効)
 - データ送信ポリシー
 - 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)



コネクタプロファイルの設定

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)

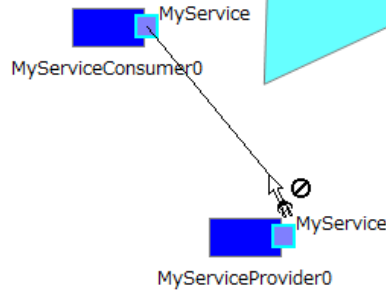


詳細にチェックを入れる

その他の項目については直接入力する

サービスポートの接続

ポートの片方からもう片方へドラッグアンドドロップ



Port Profile

ポートプロファイルを入力してください。

Name :

詳細...

Consumer	Provider

Name	Value

名前の設定

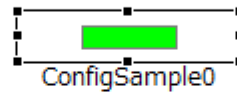
接続するインターフェースの設定
複数のインターフェースが定義されていた場合に、どのインターフェースに接続するかを設定

その他の設定を直接入力

コンフィギュレーションパラメータの設定

対象のRTCをクリックすると表示

「Configuration View」タブを選択



パラメーター一覧

ComponentName: ConfigSample0 ConfigurationSet: default

active	config	name	value
<input checked="" type="radio"/>	default	double_param0	0.99
<input type="radio"/>	mode0	double_param1	-0.99
<input type="radio"/>	mode1	int_param0	0
		int_param1	1
		str_param0	default
		str_param1	defau...
		vector_param0	0.0,0...

Buttons: 編集, 適用, キャンセル, 複製, 追加, 削除, 詳細

コンフィギュレーションセット一覧

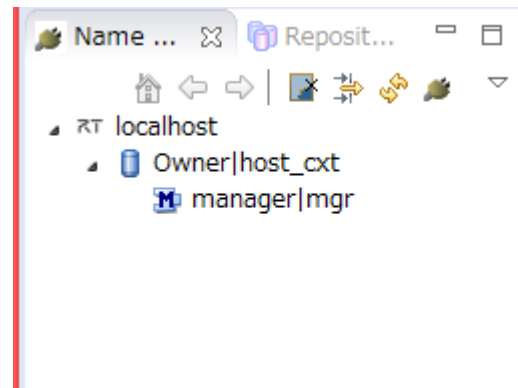
コンフィギュレーションパラメータの設定

The screenshot shows the 'Configuration' dialog box in two states. On the left, the 'ComponentName' is 'ConfigSample0' and 'ConfigurationSet' is 'default'. A red box highlights the '編集' (Edit) button. A callout bubble points to it with the text '編集ボタンを押す' (Press the edit button). A large black arrow points to the right, where the dialog box is shown in its expanded state. A red box highlights the '適用' (Apply) button. A callout bubble points to the parameter input fields with the text 'パラメータを編集する' (Edit the parameters).

This screenshot shows the 'Configuration' dialog box after editing. The 'ComponentName' is 'ConfigSample0' and 'ConfigurationSet' is 'default'. A red box highlights the '適用' (Apply) button. A callout bubble points to it with the text '適用ボタンを押す' (Press the apply button). Another callout bubble points to the parameter input fields with the text 'パラメータを編集する' (Edit the parameters).

マネージャの操作

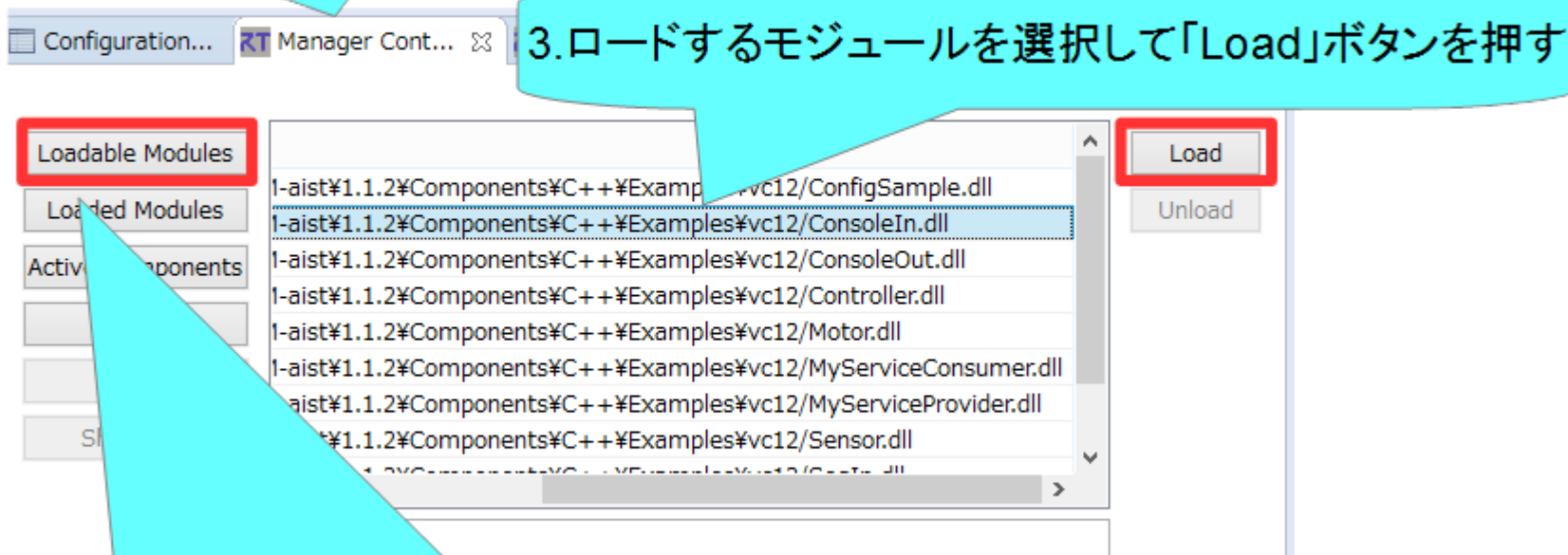
- マスターマネージャの起動、RT System Editorからの操作によるRTCの生成までの手順を説明する
 - rtc.confの設定
 - 「manager.is_master」を「YES」に設定して起動するマネージャをマスターに設定する
 - manager.is_master: YES
 - モジュール探索パスの設定
 - manager.modules.load_path: ., C:¥¥Program Files (x86)¥¥OpenRTM-aist¥¥1.1.2¥¥Components¥¥C++¥¥Examples¥¥vc12
 - 作成したrtc.confを設定ファイルの指定してrtcd.exeを起動する
 - RT Syetem Editorのネームサービスビューにマネージャが表示される



マネージャの操作

- モジュールのロード

1.「Manager Control View」タブを選択



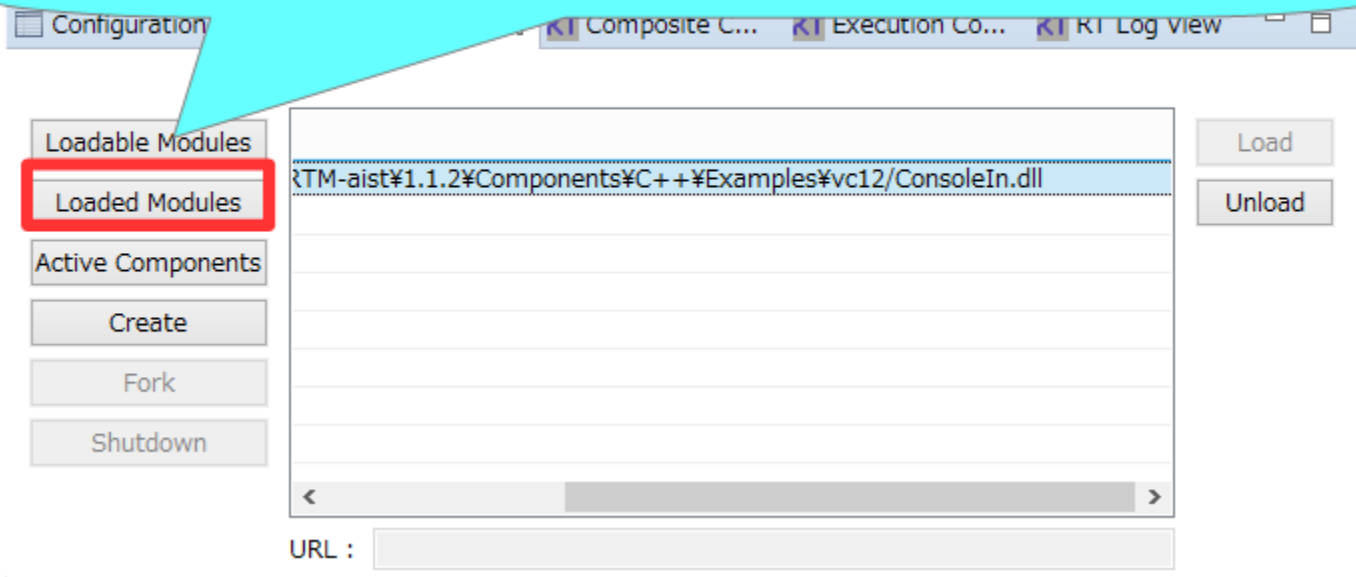
3.ロードするモジュールを選択して「Load」ボタンを押す

2.「Loadable Modules」ボタンを押すとロード可能なモジュール一覧表示

マネージャの操作

- モジュールのロード

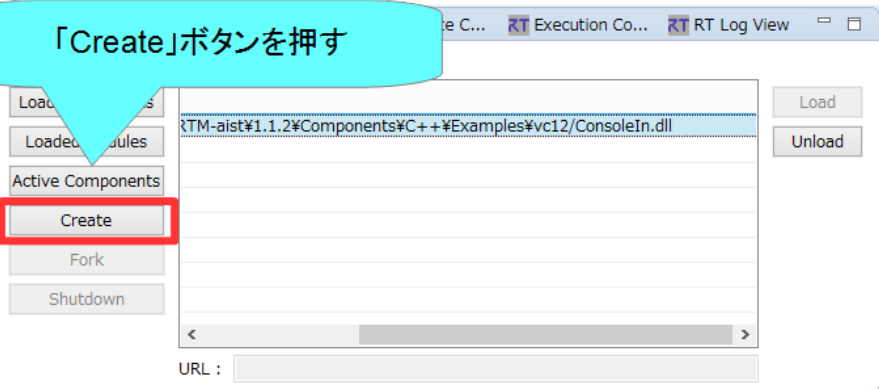
「Loaded Modules」ボタンを押すとロード済みのモジュール一覧表示



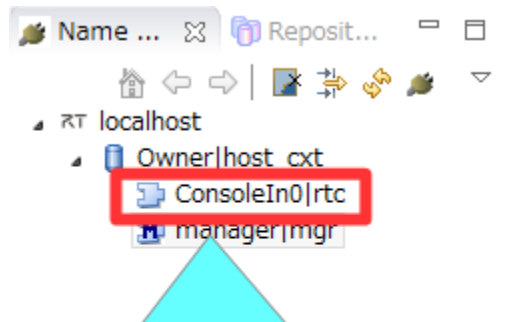
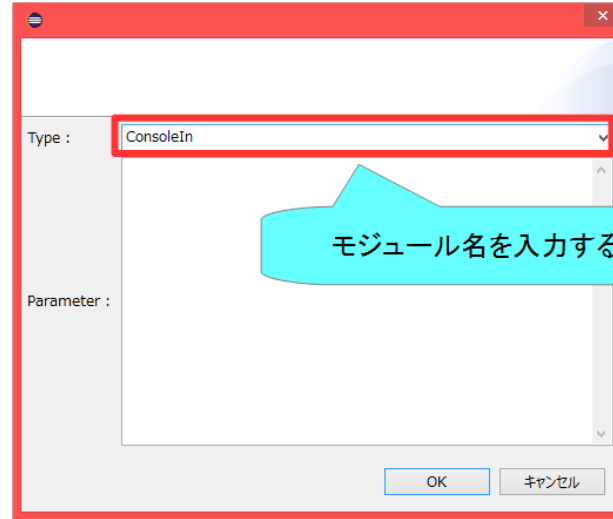
マネージャの操作

- RTCの生成

「Create」ボタンを押す



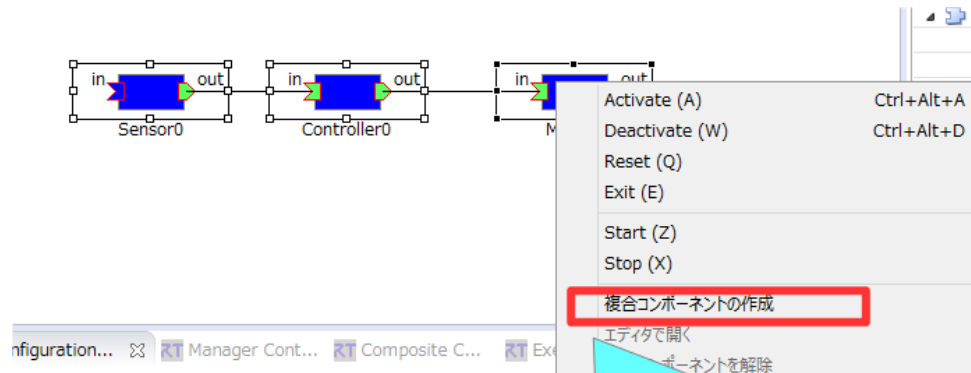
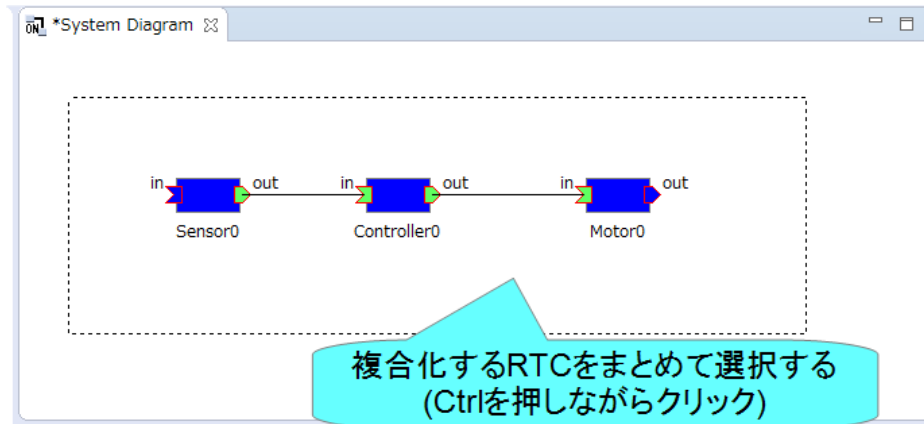
モジュール名を入力する



指定したRTCが起動する

複合コンポーネントの操作

- 複合コンポーネントの生成



右クリックして「複合コンポーネントの作成」を選択

複合コンポーネントの操作

- 複合コンポーネントの生成

The image shows a 'New Composite Component' dialog box with several callouts explaining its fields:

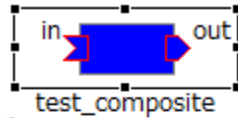
- 複合コンポーネントを起動するマネージャの指定**: Points to the 'Manager' dropdown menu.
- 名前を設定**: Points to the 'Name' text input field.
- タイプを設定**: Points to the 'Type' dropdown menu.
- 複合コンポーネントに表示するポートを指定**: Points to the 'Port' list of checkboxes.
- OKをクリックする**: Points to the 'OK' button.

To the right, a diagram shows a component named 'test_composite' with an 'in' port and an 'out' port, with a blue arrow indicating data flow between them.

複合コンポーネントが生成される: A callout pointing to the diagram.

- Type
 - 以下の3種類から選択可能
 - PeriodicECShared
 - 実行コンテキストの共有
 - PeriodicStateShared
 - 実行コンテキスト、状態の共有
 - Grouping
 - グループ化のみ

複合コンポーネントの操作



複合コンポーネントをクリック

「Composite Component View」タブを選択

Configuration... RT Manager Cont... RT Composite C... RT Execution Co... RT RT Log View

component: test_composite type: PeriodicECShared

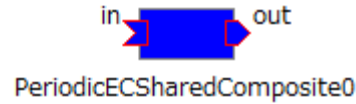
	component	port
<input type="checkbox"/>	Controller0	Controller0.in
<input type="checkbox"/>	Controller0	Controller0.out
<input checked="" type="checkbox"/>	Sensor0	Sensor0.in
<input type="checkbox"/>	Sensor0	Sensor0.out
<input type="checkbox"/>	Motor0	Motor0.in
<input checked="" type="checkbox"/>	Motor0	Motor0.out

適用
キャンセル

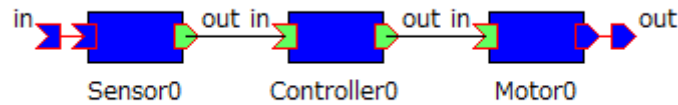
適用ボタンを押すと変更を反映

表示するポートの選択

複合コンポーネントの操作



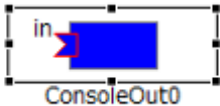
RTCをダブルクリック



子コンポーネントを表示

実行コンテキストの操作

RTCをクリック



「Execution Context View」タブを選択

実行周期

Configuration... RT Manager Cont... RT Composite C... **RT Execution Co...** RT Log View

component: ConsoleOut0

rate: 1000.0

Name	Value
id	0
kind	PERIODIC
state	RUNNING
component_state	INACTIVE
owner	ConsoleOut0
participants	0

適用
スタート
ストップ
アクティブ化
非アクティブ化
リセット
デタッチ
アタッチ

関連付けている実行コンテキスト一覧

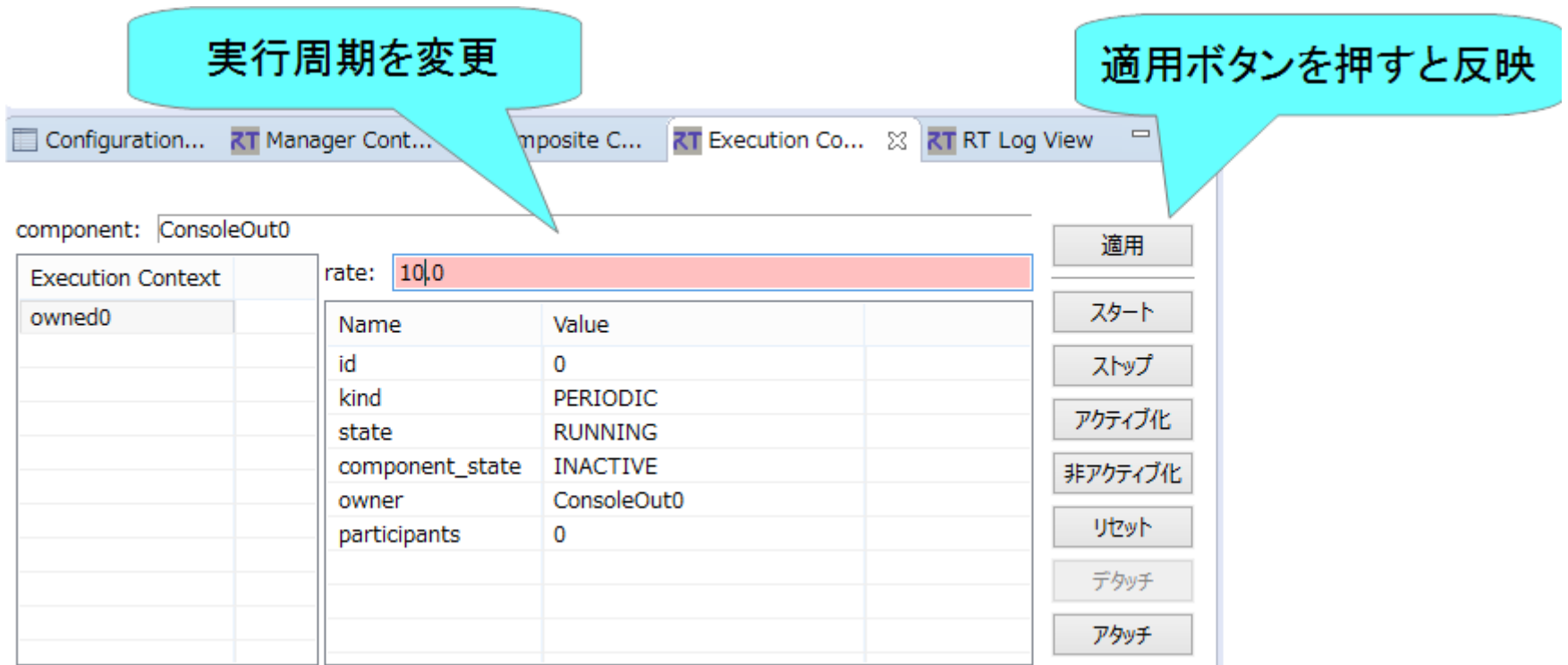
実行コンテキストの情報

実行コンテキストの操作

- 実行周期の設定

実行周期を変更

適用ボタンを押すと反映



component: ConsoleOut0

Execution Context	rate:
owned0	10.0

Name	Value
id	0
kind	PERIODIC
state	RUNNING
component_state	INACTIVE
owner	ConsoleOut0
participants	0

適用

スタート

ストップ

アクティブ化

非アクティブ化

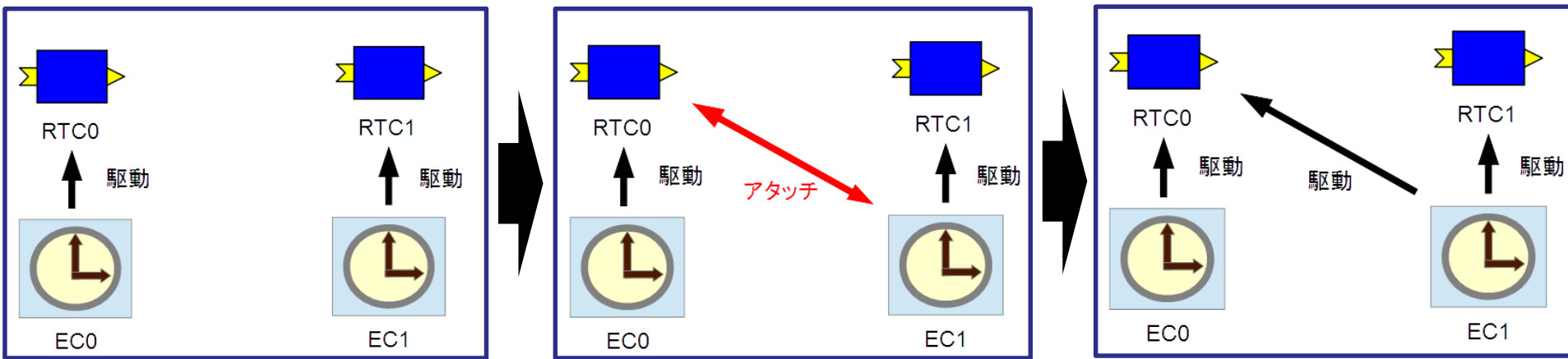
リセット

デタッチ

アタッチ

実行コンテキストの操作

- 実行コンテキストの関連付け
 - RTC起動時に生成した実行コンテキスト以外の実行コンテキストと関連付け
 - 関連付けた実行コンテキストでRTCを駆動させる
 - 他のRTCとの実行を同期させる



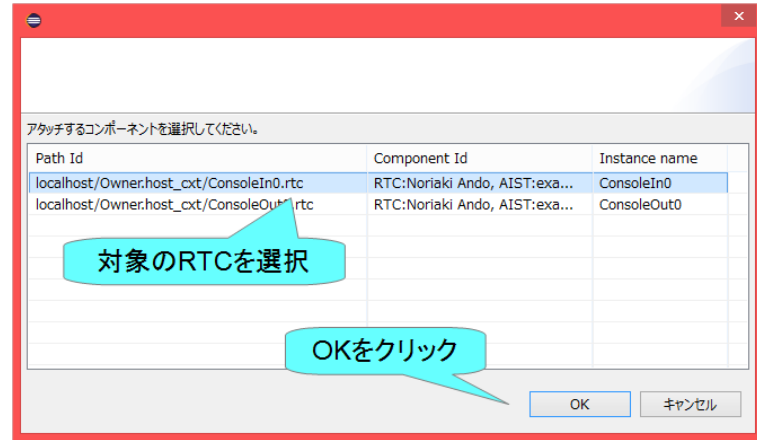
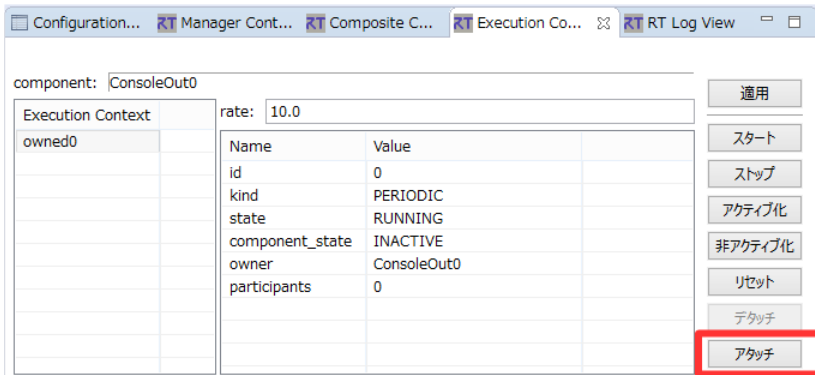
RTC0とRTC1は別々の
実行コンテキストで駆動

RTC0とEC1を関連付ける

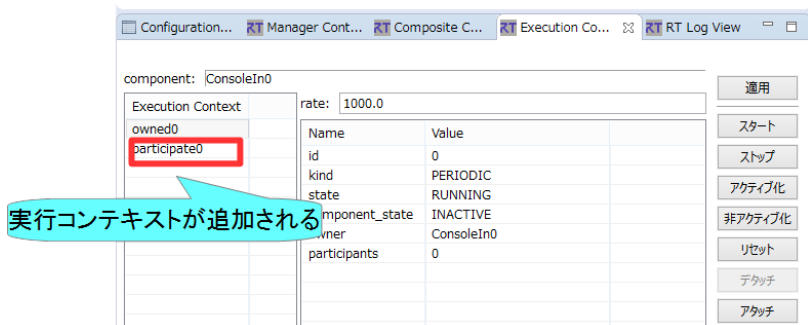
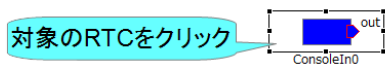
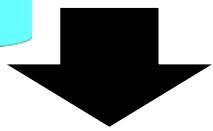
EC1がRTC0も駆動する

実行コンテキストの操作

- 実行コンテキストの関連付け

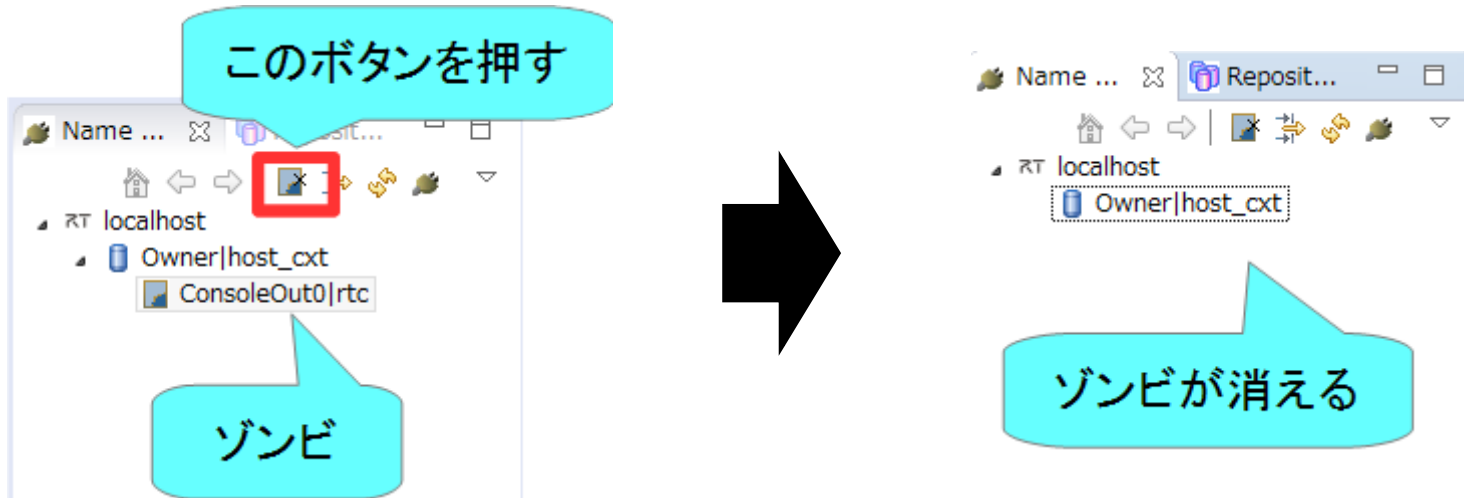


アタッチボタンを押す

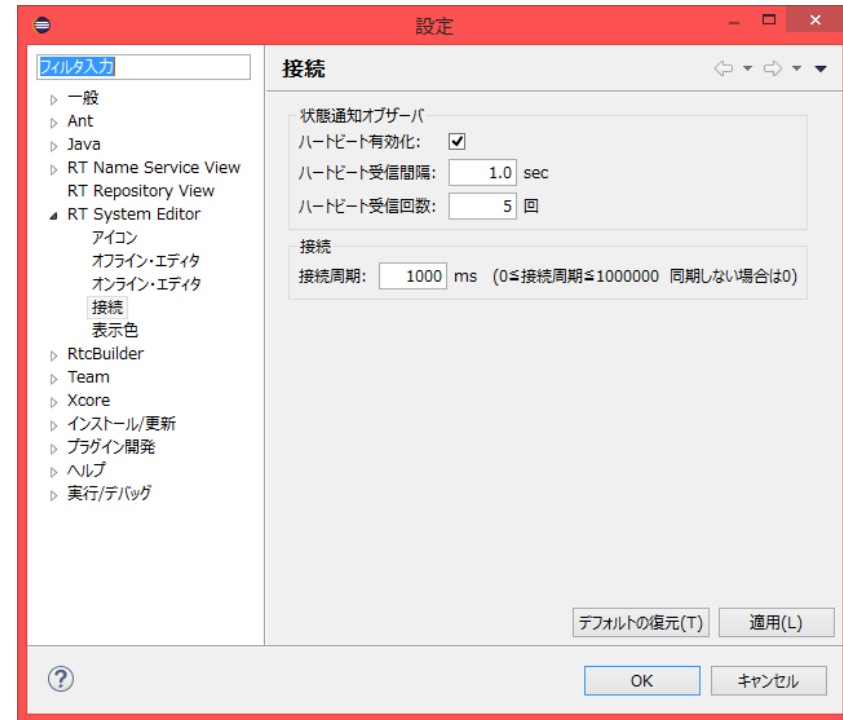
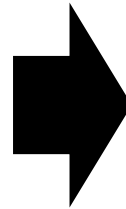
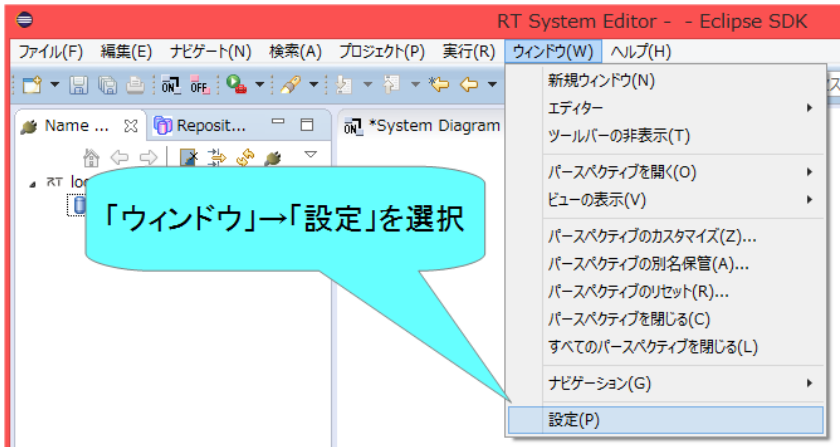


ゾンビの削除

- RTCのプロセスが異常終了する等してネームサーバーにゾンビが残った場合、以下の手順で削除する



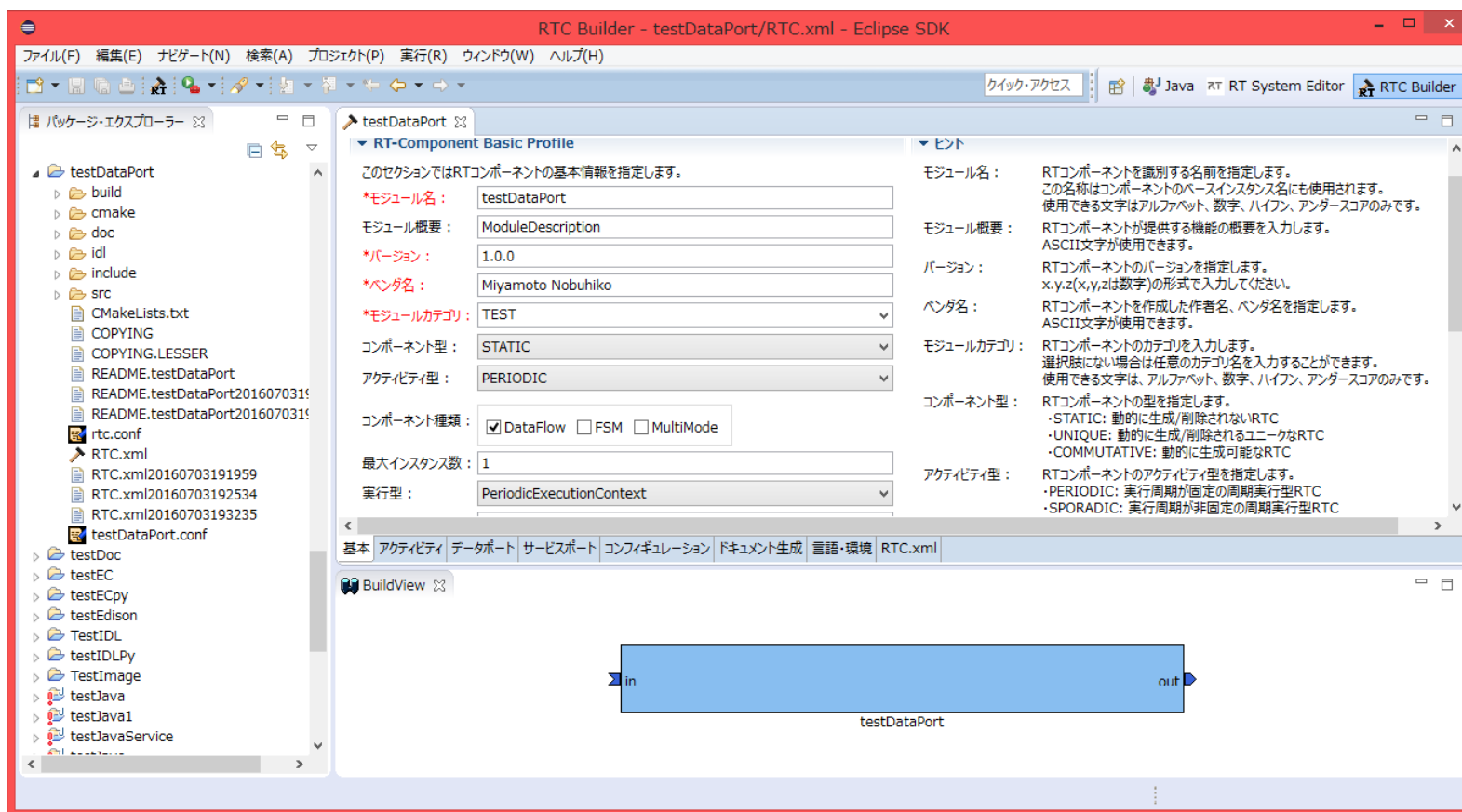
RT System Editorに関する設定



コンポーネント開発ツール RTC Builderについて

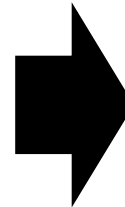
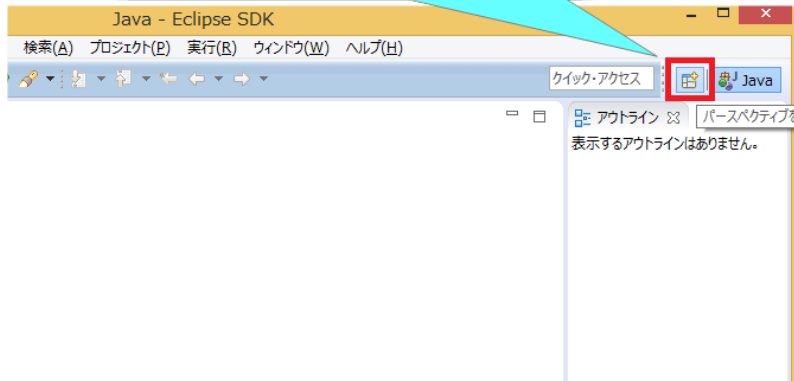
RTC Builder

- コンポーネントのプロファイル情報を入力し、ソースコード等のひな型を生成するツール
 - C++、Python、Javaのソースコードを出力

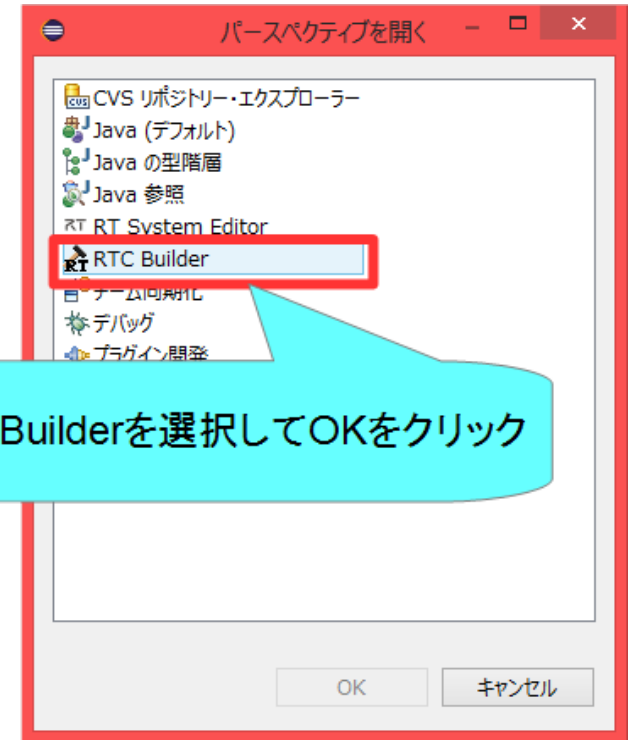


RTC Builderの起動

右上の「パースペクティブを開く」ボタンをクリック

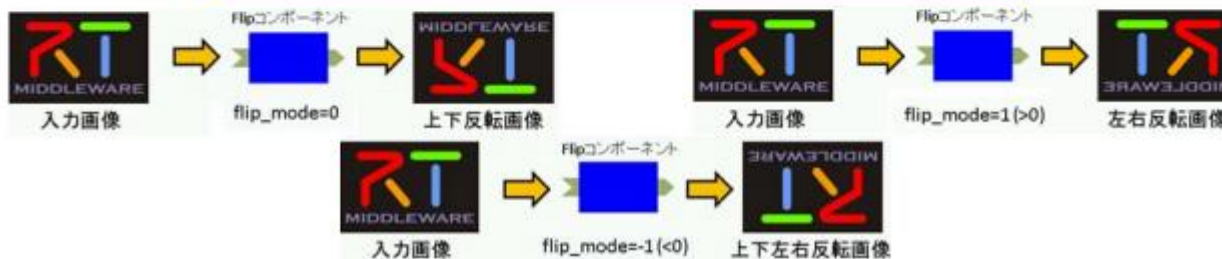
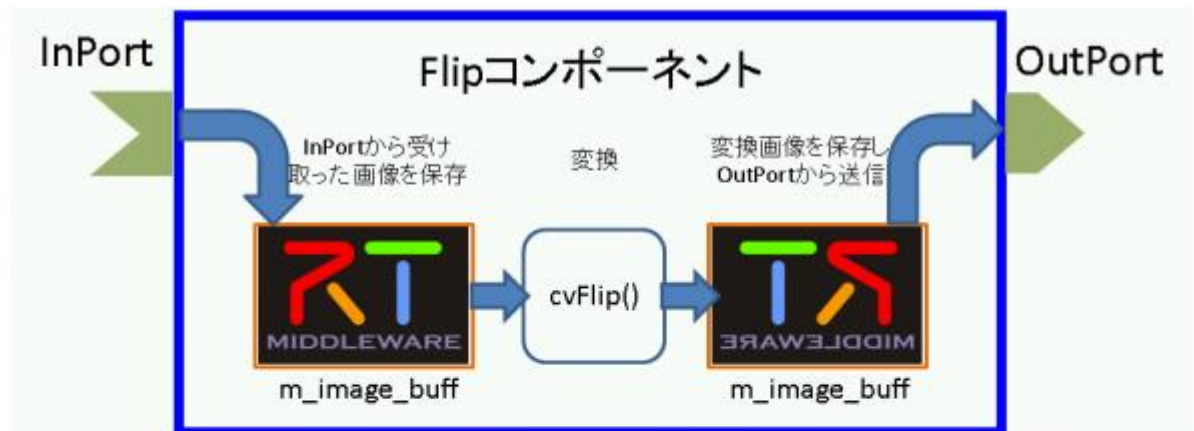


RTC Builderを選択してOKをクリック



プロジェクト作成

- 第3部プログラミング実習で使用するFlipコンポーネントのスケルトンコードを作成する。
 - 画像の反転を行うコンポーネント
 - InPortで受信した画像データを処理してOutPortから出力
 - コンフィギュレーションパラメータにより反転する方向を設定
 - RT System Editorにより他のRTCと接続、RTCをアクティブ化



第3部プログラミング実習の資料

OpenRTM-aist official website

Download OpenRTM-aist-1.1.2 for Windows (32bit, Visual C++ 2013)

現在の各言語、ツールのバージョンは以下の通りです。

- C++: 1.1.2-RELEASE
- Python: 1.1.2-RELEASE
- Java: 1.1.2-RELEASE
- tools: 1.1.2

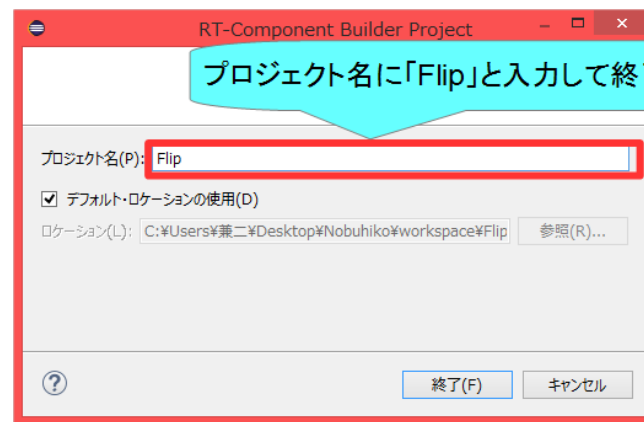
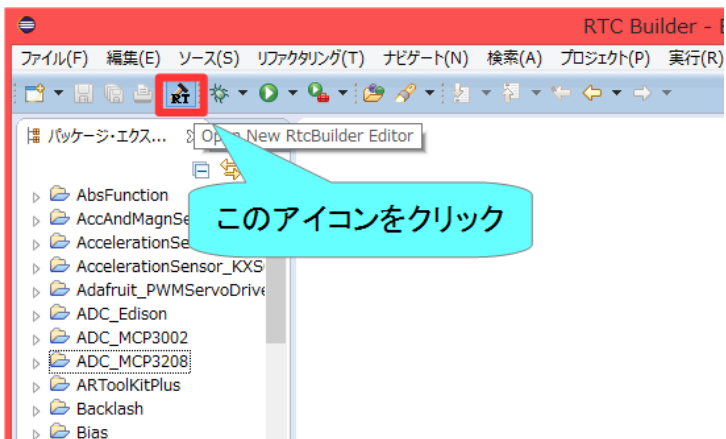


ケーススタディ

- LEGO Mindstorms EV3 活用事例
- LeapMotionでChoreonoidの制御
- Raspberry Pi Mouse 活用事例
- VPNを利用したRTMネットワーク設定方法
- GUIツールキットとRTCの連携
- **画像処理コンポーネントの作成 (Windows 8.1, OpenRTM-aist-1.1.2-RELEASE, OpenRTP-1.1.2, CMake-3.2.1, VS2013)**
- 画像処理コンポーネントの作成 (Ubuntu 16.04, OpenRTM-aist-1.1.2-RELEASE, OpenRTP-1.1.2, CMake-3.5.1, Code::Blocks-16.01)
- RTコンポーネント作成
- RTコンポーネント作成 (OpenAL用、OpenAL型の使用)
- RTコンポーネント作成 (OpenAL用、OpenAL型の使用)

「ケーススタディ」の「画像処理コンポーネントの作成」をクリック

プロジェクト作成



- Eclipse起動時にワークスペースに指定したディレクトリに「Flip」というフォルダが作成される
 - この時点では「RTC.xml」と「.project」のみが生成されている
- 以下の項目が設定する
 - 基本プロファイル
 - アクティビティ・プロファイル
 - データポート・プロファイル
 - サービスポート・プロファイル
 - コンフィギュレーション
 - ドキュメント
 - 言語環境
 - RTC.xml

基本プロフィールの入力

- 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

RTCプロフィールエディタ
ここに各項目を入力する

ヒント

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: Miyamoto Nobuhiko

*モジュールカテゴリ: TEST

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

▼ 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。選択されてい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。

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

「基本」タブを選択

基本プロファイルの入力

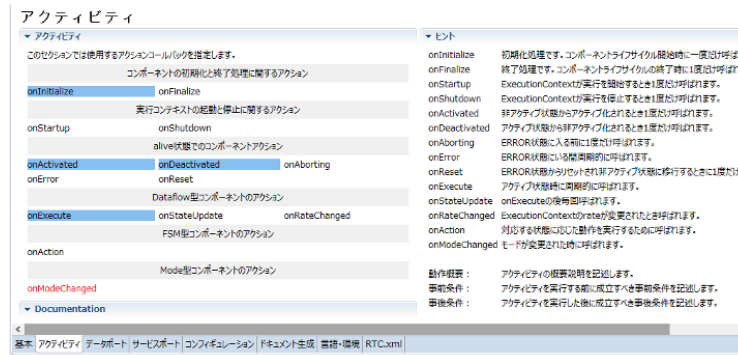
- モジュール名
 - Flip
- モジュール概要
 - 任意(Flip image component)
- バージョン
 - 任意(1.0.0)
- ベンダ名
 - 任意
- モジュールカテゴリ
 - 任意(ImageProcessing)
- コンポーネント型
 - STATIC
- アクティビティ型
 - PERIODIC
- コンポーネントの種類
 - DataFlow
- 最大インスタンス数
 - 1
- 実行型
 - PeriodicExecutionContext
- 実行周期
 - 1000.0
- 概要
 - 任意

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 :	ImageProcessing
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 :	<input checked="" type="checkbox"/> DataFlow <input type="checkbox"/> FSM <input type="checkbox"/> MultiMode
最大インスタンス数 :	1
実行型 :	PeriodicExecutionContext
実行周期 :	1000.0
概要 :	
RTC Type :	

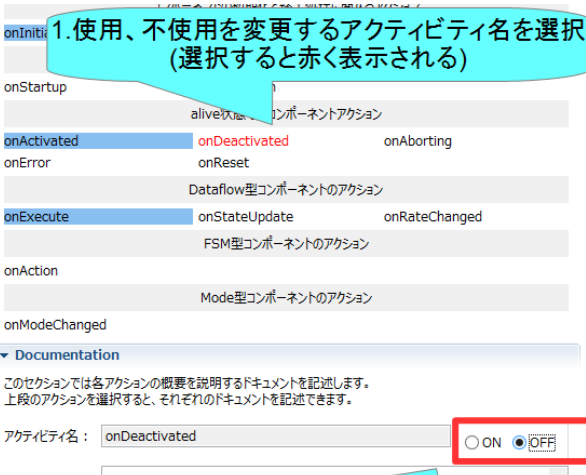
アクティビティの設定

- 使用するアクティビティを設定する

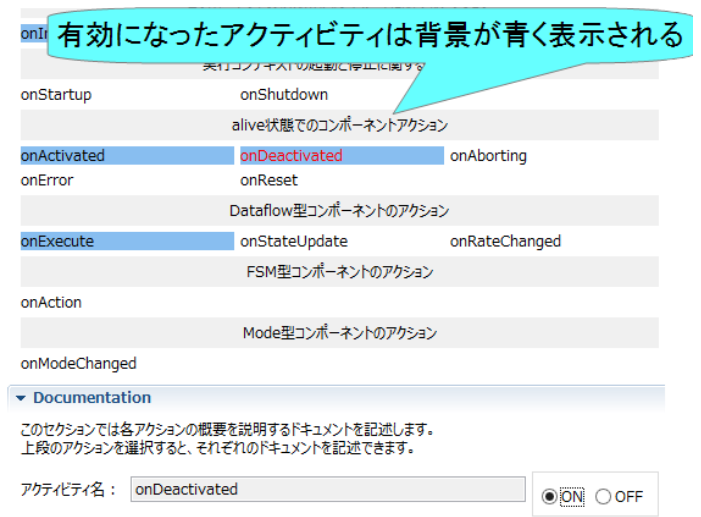


「アクティビティ」タブを選択

- 指定アクティビティを有効にする手順



2. アクティビティ名の選択後、ON・OFFを選択する



アクティビティの設定

- 以下のアクティビティを有効にする
 - onInitialize
 - onActivated
 - onDeactivated
 - onExecute
- Documentationは適当に書いておいてください
 - 空白でも大丈夫です

▼ アクティビティ

このセクションでは使用するアクションコールバックを指定します。

コンポーネントの初期化と終了処理に関するアクション

onInitialize onFinalize

実行コンテキストの起動と停止に関するアクション

onStartup onShutdown

alive状態でのコンポーネントアクション

onActivated onDeactivated onAborting

onError onReset

Dataflow型コンポーネントのアクション

onExecute onStateUpdate onRateChanged

FSM型コンポーネントのアクション

onAction

Mode型コンポーネントのアクション

onModeChanged

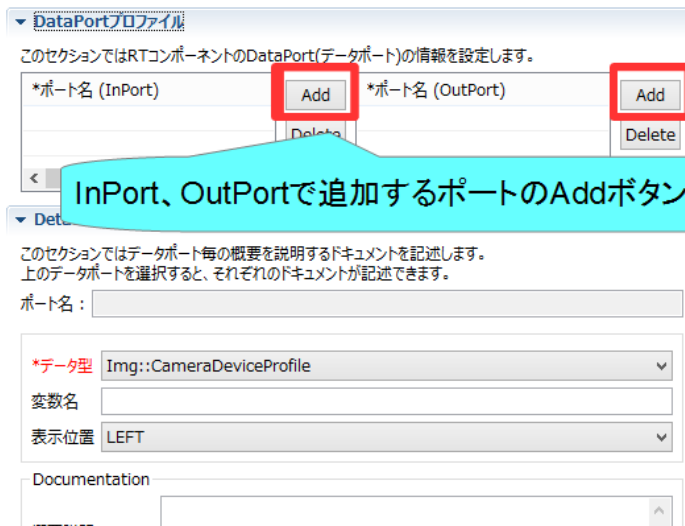
データポートの設定

- InPort、OutPortの追加、設定を行う

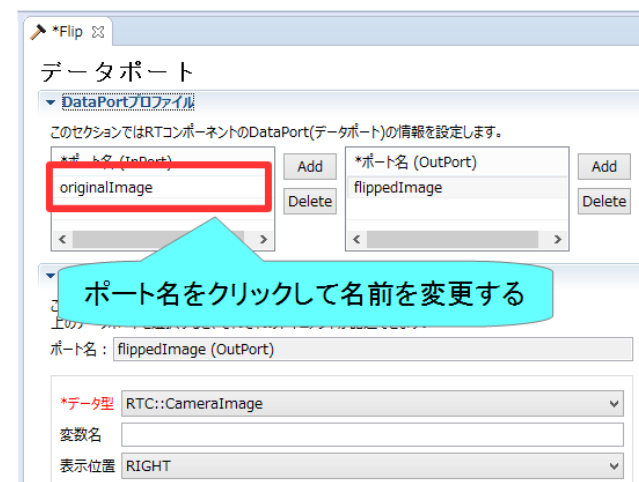
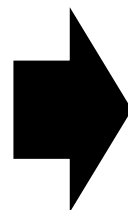


「データポート」タブを選択

データポートを追加する手順



InPort、OutPortで追加するポートのAddボタンをクリック

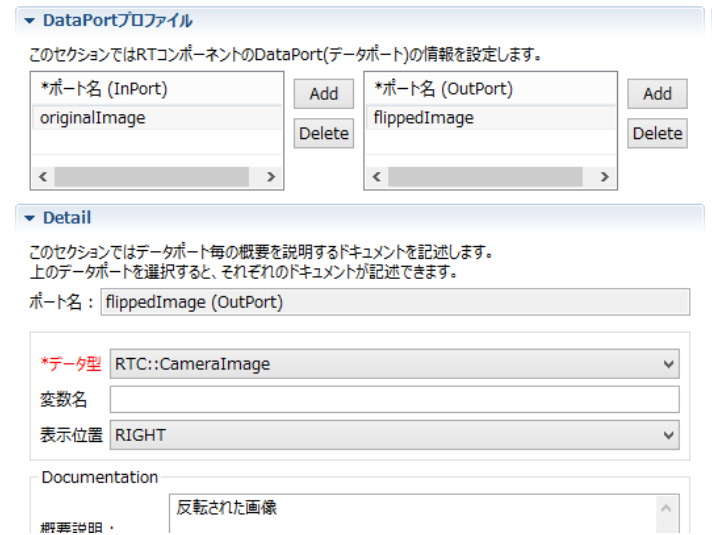


ポート名をクリックして名前を変更する

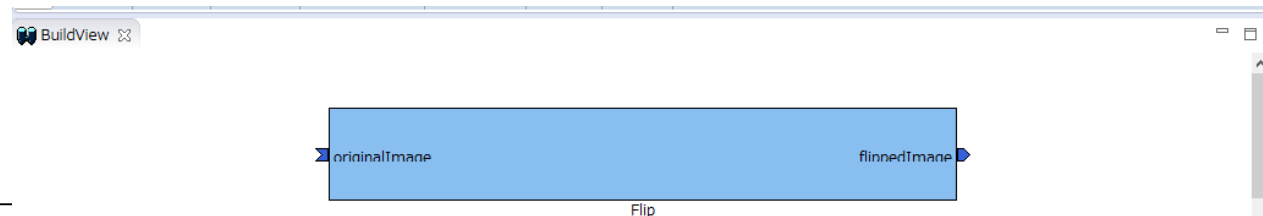
各項目を設定する

データポートの設定

- 以下のInPortを設定する
 - originalImage
 - データ型: RTC::CameraImage
 - 他の項目は任意
- 以下のOutPortを設定する
 - flippedImage
 - データ型: RTC::CameraImage
 - 他の項目は任意

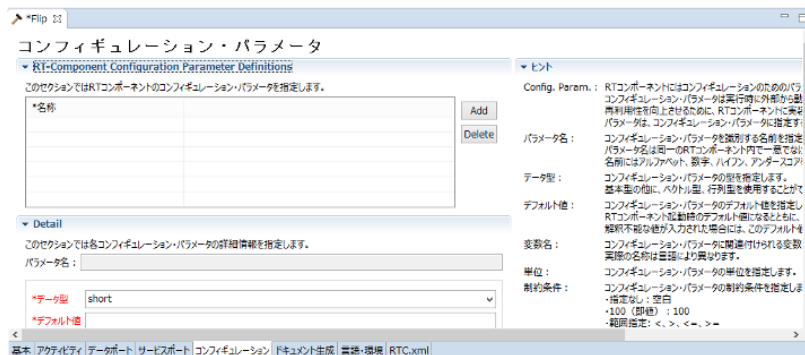


- ※今回使用するのは **RTC::CameraImage**なので **Img::CameraImage**と間違えないようにする。

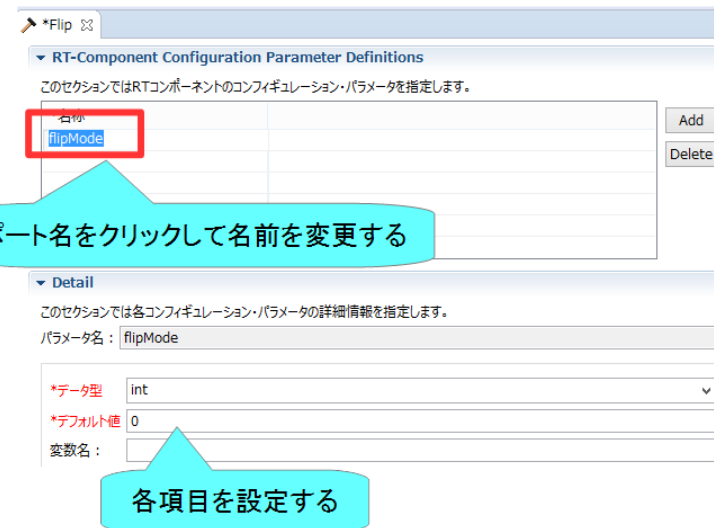
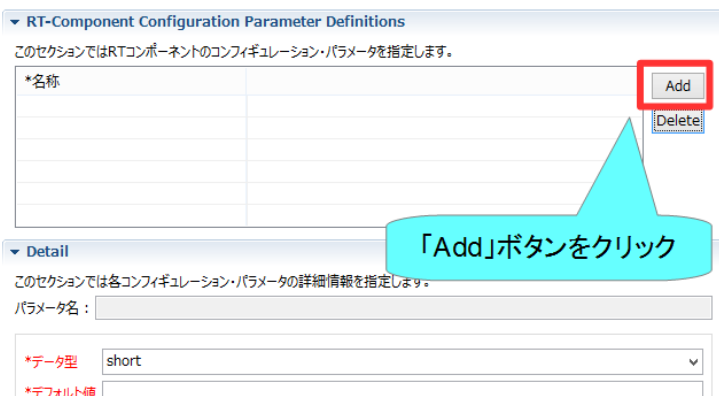


コンフィギュレーションの設定

- コンフィギュレーションパラメータの追加、設定を行う



- コンフィギュレーションパラメータを追加する手順



コンフィギュレーションの設定

- 以下のコンフィギュレーションパラメータを設定する
 - flipMode
 - データ型: int
 - デフォルト値: 0
 - 制約条件: (0,-1,1)
 - Widget: radio
 - 他の項目は任意

*名称		Add
flipMode		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

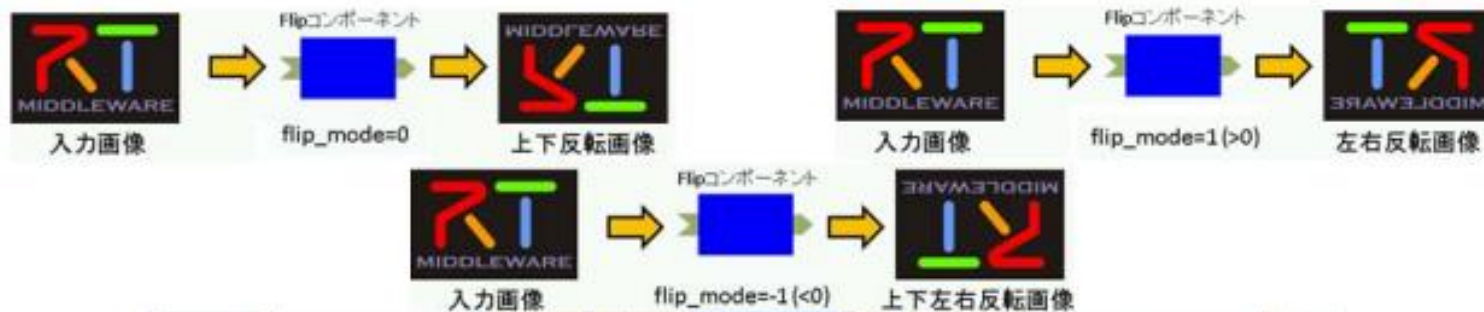
変数名: _____

単位: _____

制約条件: (0,-1,1)

Widget: radio

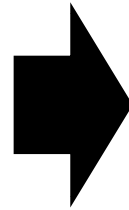
Step: _____



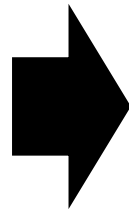
コンフィギュレーションパラメータの制約、Widgetの設定

- RT System Editorでコンフィギュレーションパラメータを編集する際にGUIを表示する

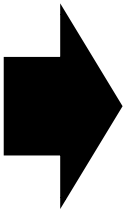
- Widget: text



- 制約条件: $0 \leq x \leq 100$
- Widget: slider
- Step: 10

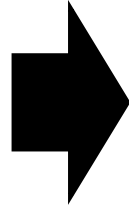


- 制約条件: $0 \leq x \leq 100$
- Widget: spin
- Step: 10



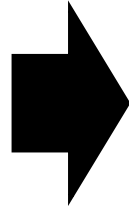
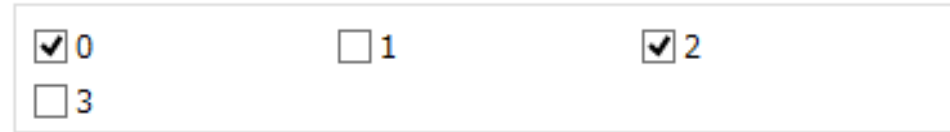
コンフィギュレーションパラメータの制約、Widgetの設定

- 制約条件: (0,1,2,3)
- Widget: radio



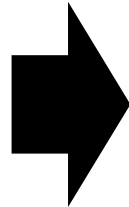
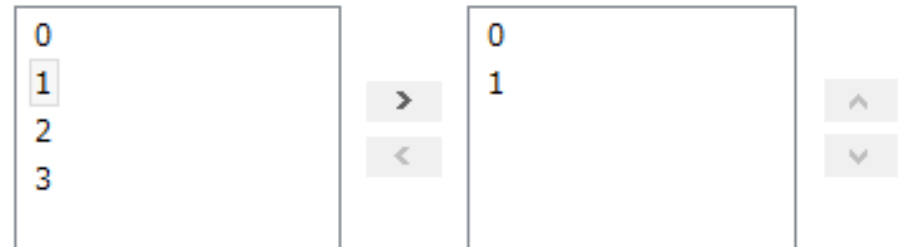

○ 0 ○ 1 ● 2
○ 3

- 制約条件: (0,1,2,3)
- Widget: checkbox

☑ 0 ☐ 1 ☑ 2
☐ 3

- 制約条件: (0,1,2,3)
- Widget: ordered_list

0
1
2
3

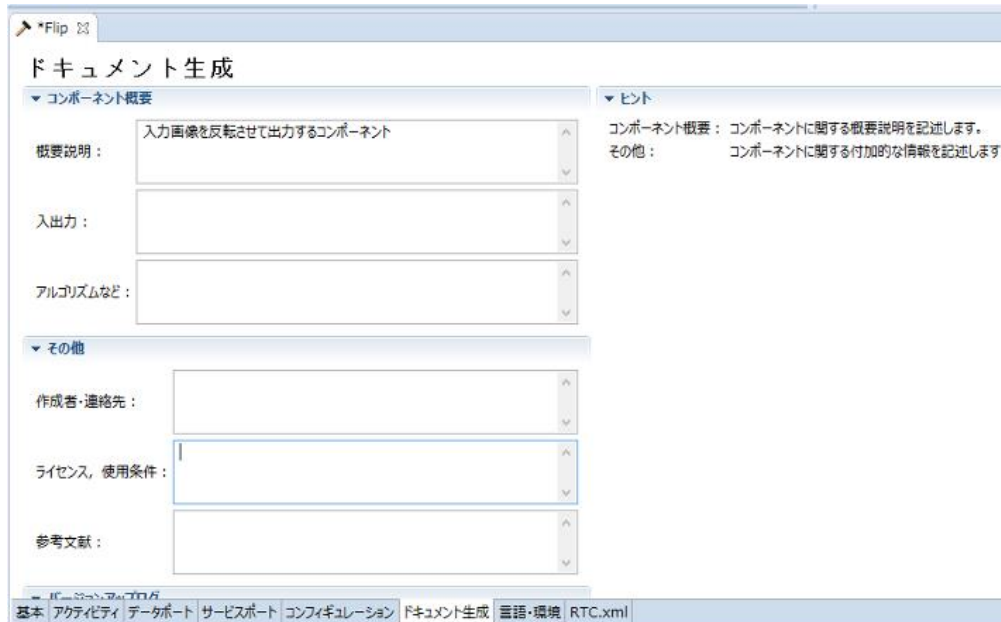
> <

0
1

^ v

ドキュメントの設定

- 各種ドキュメント情報を設定

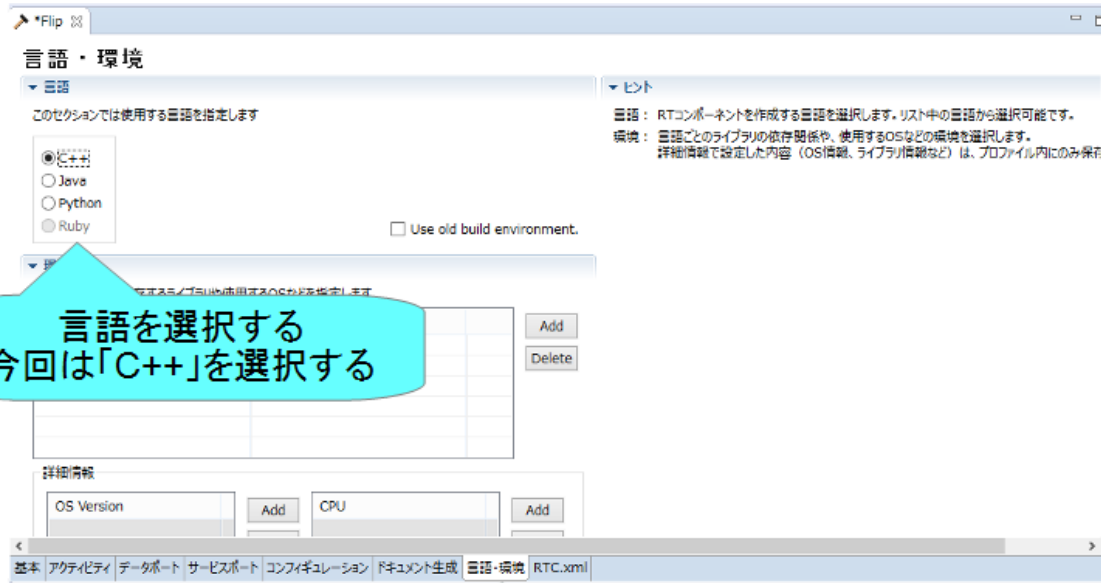


「ドキュメント生成」タブを選択

- 今回は適当に設定しておいてください。
 - 空白でも大丈夫です

言語の設定

- 実装する言語，動作環境に関する情報を設定

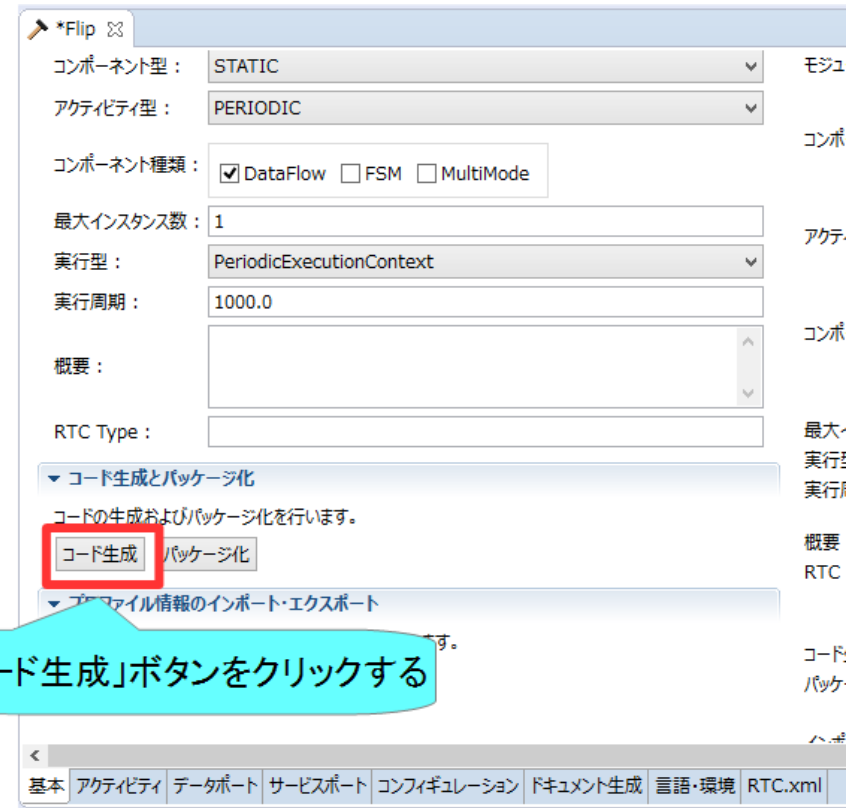


言語を選択する
今回は「C++」を選択する

「言語・環境」タブを選択

スケルトンコードの生成

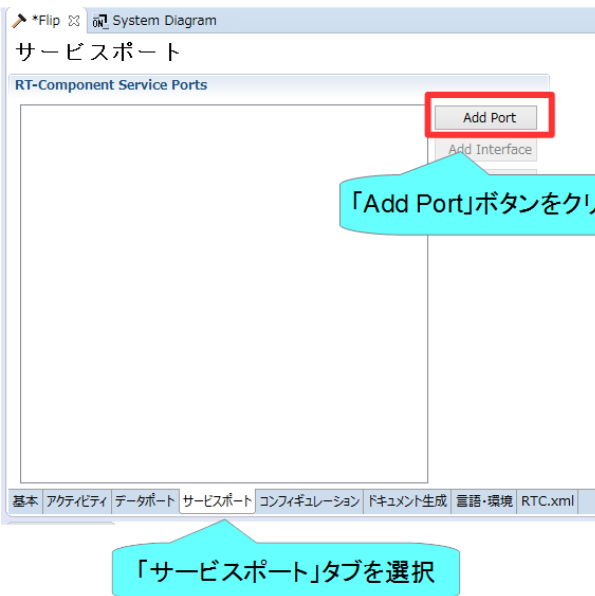
- 基本タブからコード生成ボタンを押すことでスケルトンコードが生成される
 - ソースコード
 - C++ソースファイル(.cpp)
 - ヘッダーファイル(.h)
 - このソースコードに画像を反転させる処理を記述する
 - CMakeの設定ファイル
 - CMakeLists.txt
 - 以下略



RTC Builder 補足

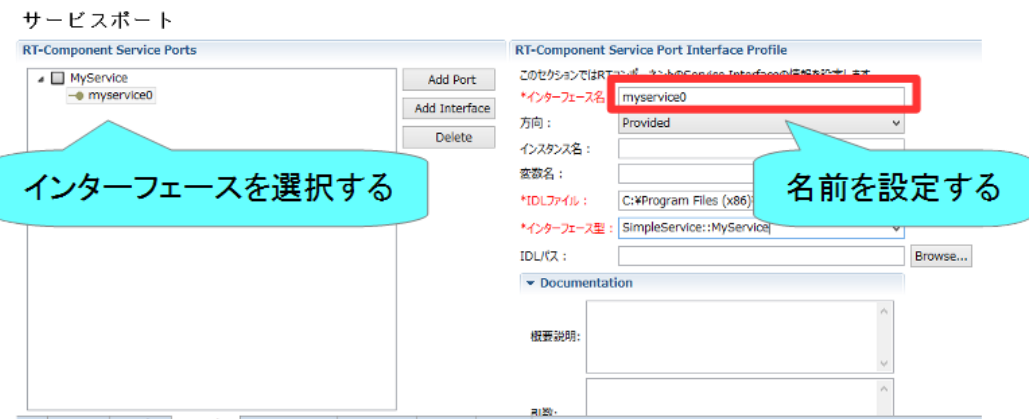
サービスポートの設定

- サービスポートの追加、インターフェースの追加、設定を行う



サービスポートの設定

- インターフェースを追加する



サービスポートの設定

- インターフェースの設定を行う

このセクションではRTコンポーネントのService Interfaceの情報を設定

*インターフェース名 : myservice0

方向 : Provided

インスタンス名 :

変数名 :

*IDLファイル : C:\Program Files (x86)\OpenRTM-aist\1.1.2\Com Browse...

*インターフェース型 : SimpleService::MyService

IDLパス : Browse...

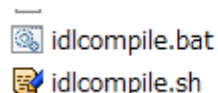
「Provided」・「Required」から選択
 Provided: サービスを提供する側
 Required: サービスを利用する側

「Browse...」をクリックしてIDLファイルを選択

インターフェース型を選択

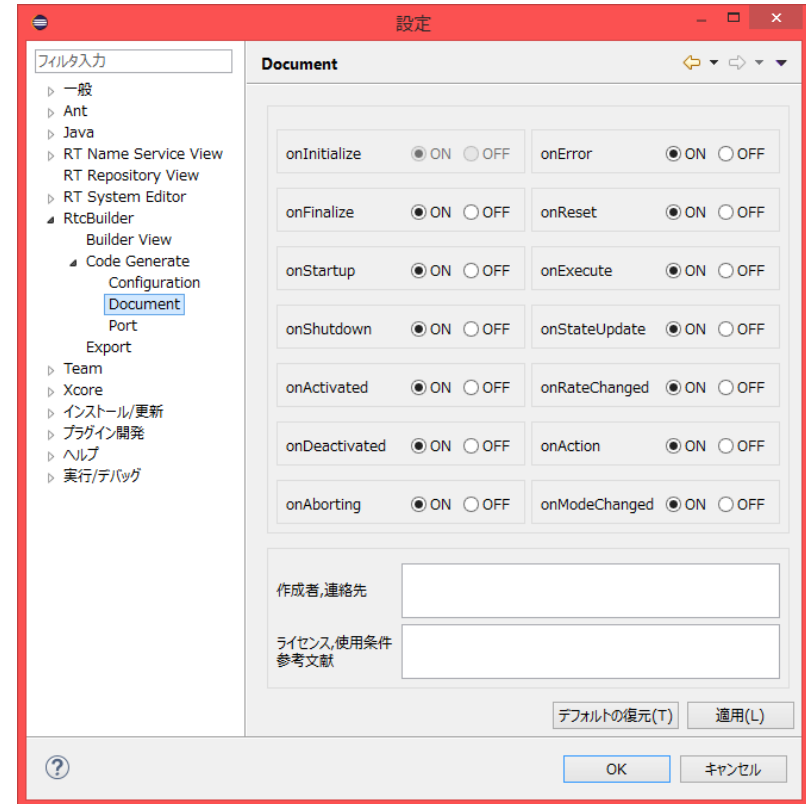
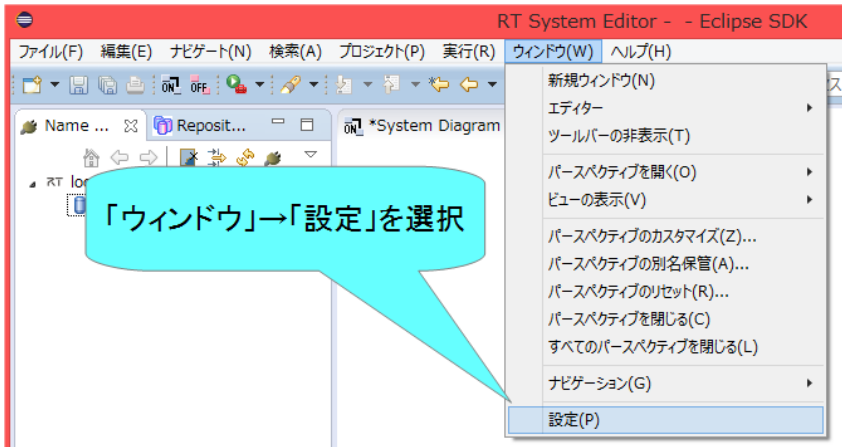
IDLファイルが別のIDLファイルをインクルードしている場合にIDLパスを設定

- コード生成後、Pythonの場合は
idlcompile.bat(idlcompile.sh)を起動する



2016/07/03 18:07 Windows J
 2016/07/03 18:07 SH ファイル

RTC Builderに関する設定



RTC Builderに関する設定

設定

フィルタ入力

- 一般
- Ant
- Java
- RT Name Service View
- RT Repository View
- RT System Editor
- RtcBuilder
 - Builder View
 - Code Generate
 - Configuration
 - Document**
 - Port

Document

onInitialize	<input checked="" type="radio"/> ON <input type="radio"/> OFF	onError	<input type="radio"/> ON <input checked="" type="radio"/> OFF
onFinalize	<input type="radio"/> ON <input checked="" type="radio"/> OFF	onReset	<input type="radio"/> ON <input checked="" type="radio"/> OFF
onStartup	<input type="radio"/> ON <input checked="" type="radio"/> OFF	onExecute	<input checked="" type="radio"/> ON <input type="radio"/> OFF
onShutdown	<input type="radio"/> ON <input checked="" type="radio"/> OFF	onStateUpdate	<input type="radio"/> ON <input checked="" type="radio"/> OFF
onActivated	<input checked="" type="radio"/> ON <input type="radio"/> OFF	onRateChanged	<input type="radio"/> ON <input checked="" type="radio"/> OFF
onDeactivated	<input checked="" type="radio"/> ON <input type="radio"/> OFF	onAction	<input type="radio"/> ON <input checked="" type="radio"/> OFF
onAborting	<input type="radio"/> ON <input checked="" type="radio"/> OFF	onModeChanged	<input type="radio"/> ON <input checked="" type="radio"/> OFF

作成者,連絡先: Nobuhiko Miyamoto <n-miyamoto@aist.go.jp>

ライセンス,使用条件,参考文献: LGPL

適用(T) 適用(L)

「RtcBuilder」→「Code Generate」→「Document」を選択

「onActivated」、「onDeactivated」、「onExecute」はよく使うので「ON」にしておくとプロジェクトを新規作成したときに自動的にONになるので作業が減る。

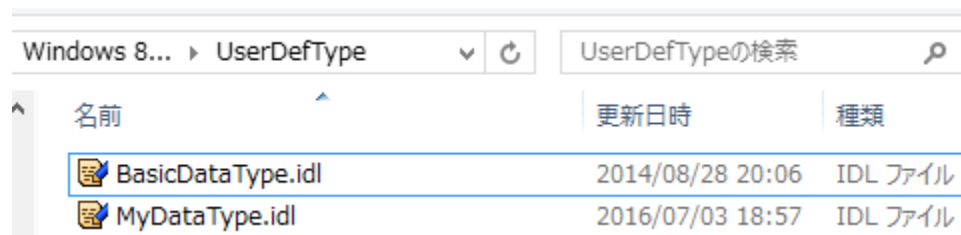
「作成者、連絡先」、「ライセンス、使用条件、参考文献」を入力しておくともプロジェクト作成時に自動的に入力されるので便利。

独自のデータ型の利用

- 独自のデータ型でデータポートの通信を行う手順
 - IDLファイルを作成する
 - MyDataType.idlを任意のフォルダ(ここではC:¥UserDefType)作成

```
1 // @file MyDataType.idl ↓
2 #include "BasicDataType.idl" ↓
3 ↓
4 struct MyData ↓
5 { ↓
6     RTC::Time tm; ↓
7     short shortVariable; ↓
8     long longVariable; ↓
9     sequence<double> data; ↓
10 }; [EOF]
```

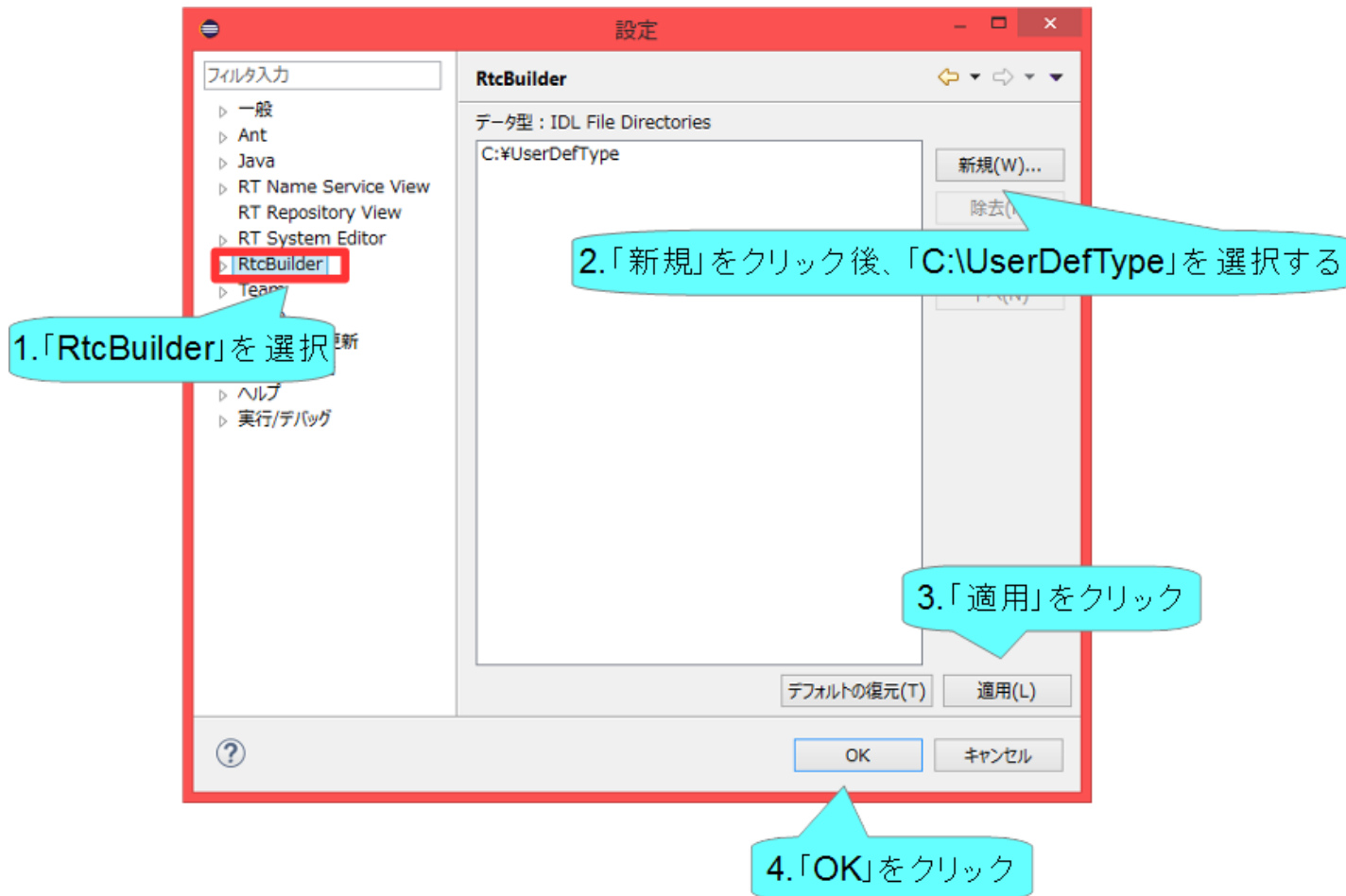
- 別のIDLファイルをインクルードしている場合は同じフォルダにコピーする



名前	更新日時	種類
BasicDataType.idl	2014/08/28 20:06	IDL ファイル
MyDataType.idl	2016/07/03 18:57	IDL ファイル

独自のデータ型の利用

- 独自のデータ型でデータポートの通信を行う手順
 - RTC Builderの設定でIDLファイルの存在するディレクトリを追加



独自のデータ型の利用

- 独自のデータ型でデータポートの通信を行う手順

▼ DataPortプロフィール

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

<p>*ポート名 (InPort)</p> <table style="width: 100%; border-collapse: collapse;"> <tr><td style="border-bottom: 1px solid #ccc; padding: 2px;">in</td></tr> <tr><td style="border-bottom: 1px solid #ccc; padding: 2px;"> </td></tr> <tr><td style="border-bottom: 1px solid #ccc; padding: 2px;"> </td></tr> </table> <p style="text-align: right;">Add Delete</p>	in			<p>*ポート名 (OutPort)</p> <table style="width: 100%; border-collapse: collapse;"> <tr><td style="border-bottom: 1px solid #ccc; padding: 2px;">out</td></tr> <tr><td style="border-bottom: 1px solid #ccc; padding: 2px;"> </td></tr> <tr><td style="border-bottom: 1px solid #ccc; padding: 2px;"> </td></tr> </table> <p style="text-align: right;">Add Delete</p>	out		
in							
out							

▼ 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:

*データ型	<div style="background-color: #e6f2ff; padding: 2px;">MyData</div>
変数名	MyData
表示位置	RTC::Acceleration2D
Document	RTC::Acceleration3D
	RTC::ActuatorCurrent
	RTC::ActuatorGeometry

データ型一覧にMyDataが追加