

RTミドルウェア サマーキャンプ2011

日時:2011年8月29日(月)~9月2日(金)

場所:産業技術総合研究所 中央第2 本部情報棟1F 交流会議室



13:00- 13:15	実行委員長挨拶と本講習会の趣旨, スケジュールの説明 担当: 神徳 徹雄 (産業技術総合研究所)
13:15- 14:05	参加者自己紹介と本講習会でのモチベーションについて 各参加者
14:20- 15:30	OpenRTM-aist-1.1について 担当: 栗原真二 (産業技術総合研究所)
15:40- 17:00	特別講演1: RTCBuilderを用いたサンプルコンポーネントの作成および RTSystemEditorを用いたコンポーネントの動作確認 担当: 坂本武志 (株式会社グローバルアシスト)
17:00- 17:15	さくら館部屋割りと産総研周囲の食事処などの紹介 担当: 尾形仁子 (産業技術総合研究所)

特別講演1

RTCBuilderを用いたサンプルコンポーネントの作成 およびRTSystemEditorを用いたコンポーネントの動作確認

株式会社 グローバルアシスト
坂本 武志

■ 次世代ロボット知能ソフトウェアプラットフォーム

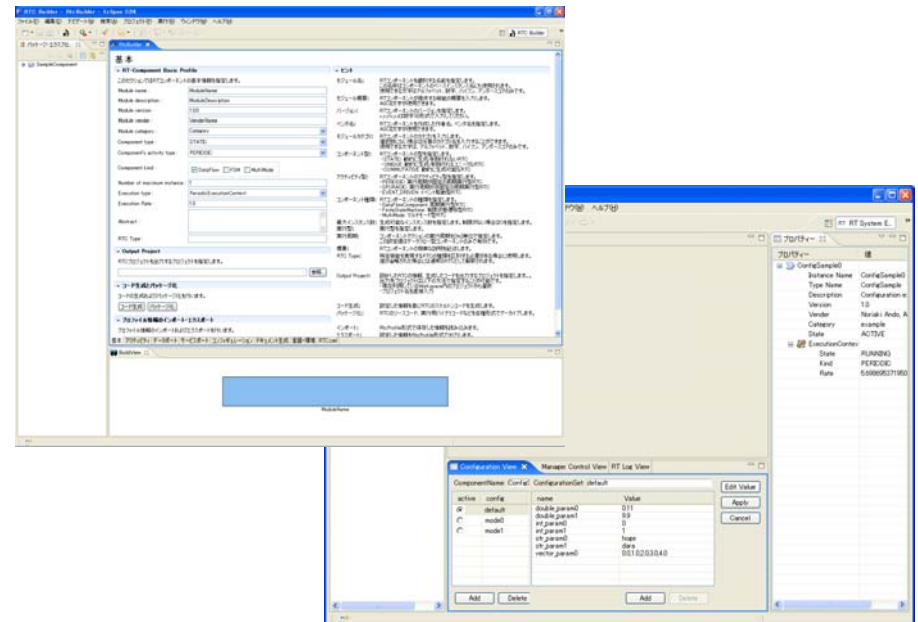
- <http://www.openrtp.jp/wiki/>
- システム設計, シミュレーション, 動作生成, シナリオ生成などをサポート

■ OpenRT Platformツール群

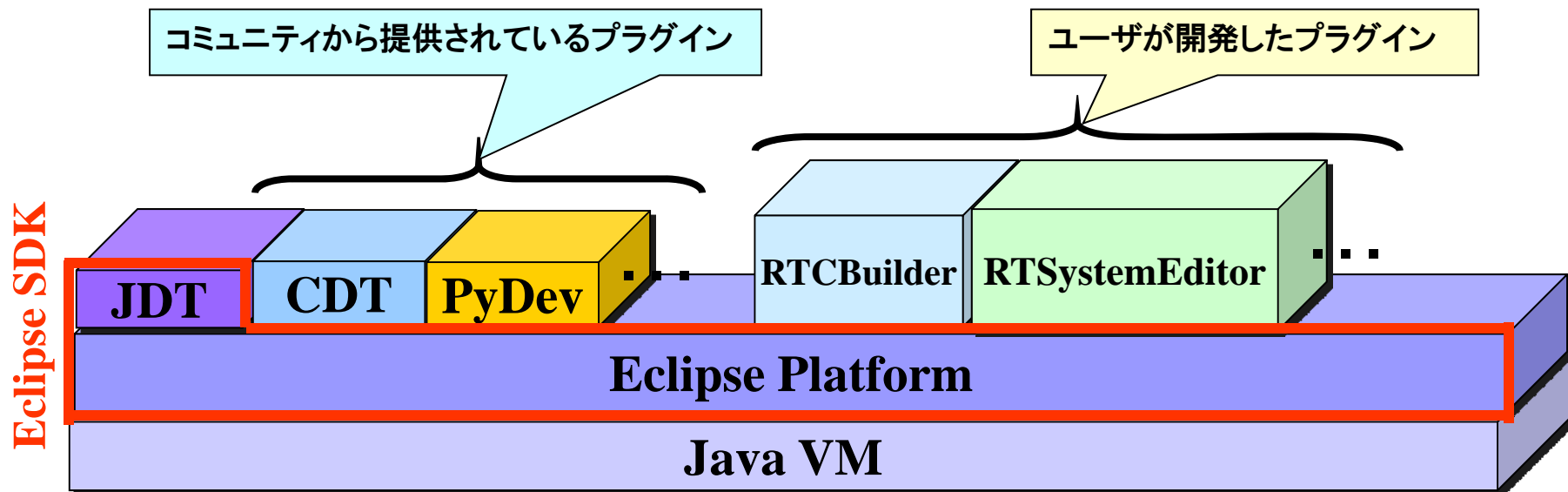
- コンポーネント開発, システム開発における各開発フェーズの作業支援
- 開発プラットフォームにEclipseを採用

■ 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



- 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



■ 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

The screenshot shows the OpenRTM-aist website. The main content area is titled "OpenRTM Eclipse tools 1.0-RELEASE". It includes a navigation menu on the left, a "Table of contents" section, and a table of download links for various operating systems and languages.

Home - ダウンロード - ツール - Eclipse tools 1.0-RELEASE

OpenRTM Eclipse tools 1.0-RELEASE

これまで、OpenRTM-aistのツールとして開発されてきた RTCBuilder (旧RtcTemplate) および RTSystemEditor (旧 RtcLink) は、OpenHRP3やその他のツールと統合開発環境を構成する OpenRT Platform に組み込まれることになりました。こちらでは、RTSystemEditor 及び RTCBuilder のみを配布していますが、将来的には様々なツールを一括で提供する予定です。

現在の RTSystemEditor 及び RTCBuilder の最新バージョンは 1.0.0 です。

Table of contents

- 全部入りパッケージ
- バイナリ
- RTSystemEditor/RTCBuilderデベロッパービルド
- Eclipse/JDK/JRE等
- 過去のバージョン

全部入りパッケージ

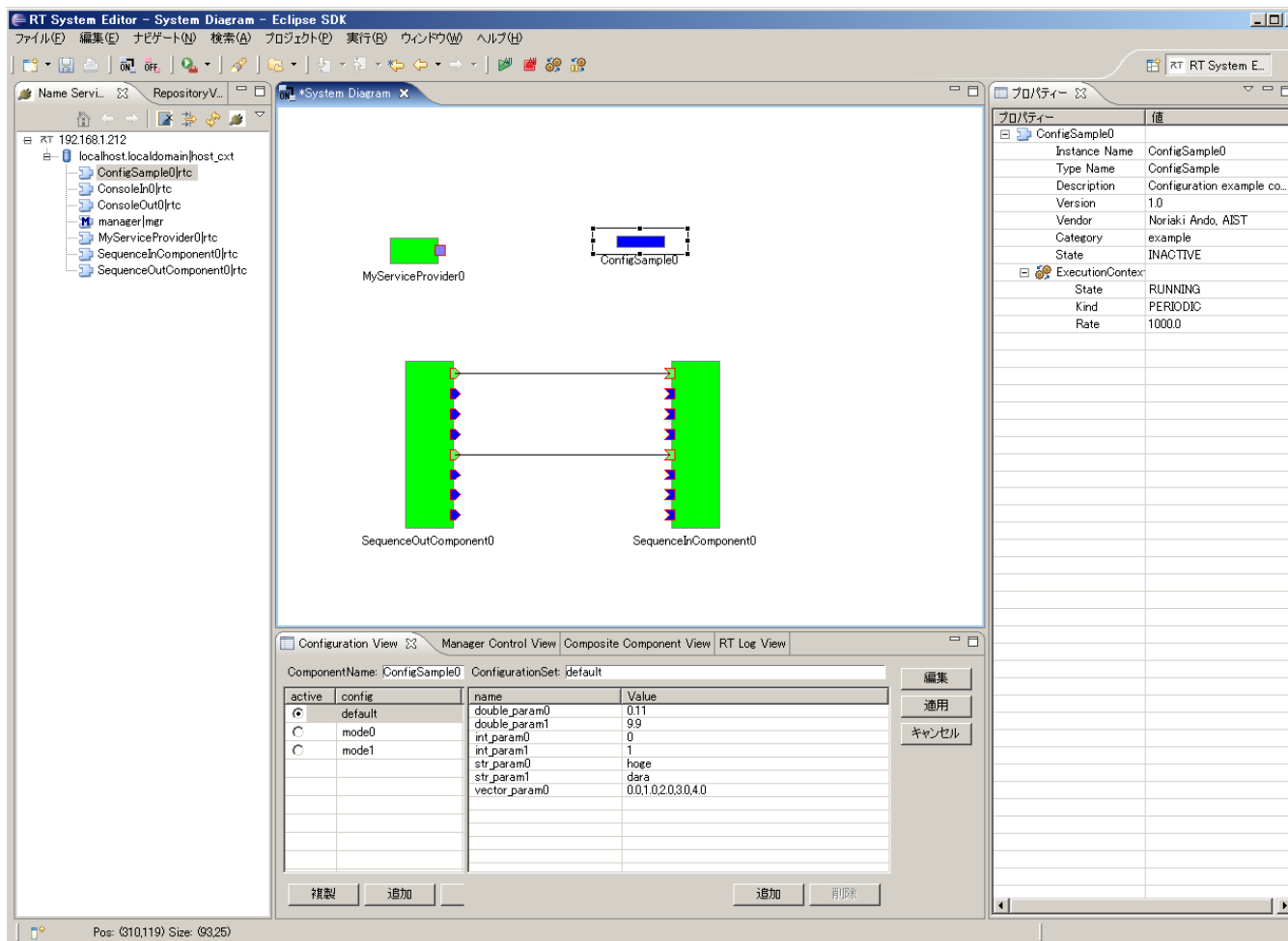
Eclipse-3.4.2 [Ganymede SR2]		
Eclipse3.4.2+RTSE+RTCB Windows用全部入り	eclipse342_rtmttools100release_win32_ja.zip MD5:A52450B24F0A1C59402D5340D9FA8D56	2010.06.01
Eclipse3.4.2+RTSE+RTCB (英語版) Windows用全部入り	eclipse342_rtmttools100release_win32_en.zip MD5:2A1895F0E01D874E35CDD29EFDCE1DE7	2010.06.01
Eclipse3.4.2+RTSE+RTCB Linux用全部入り	eclipse342_rtmttools100release_linux_ja.tar.gz MD5:FD54B638BB72A351D92ACD22CD4099C7	2010.06.01
Eclipse3.4.2+RTSE+RTCB (英語版) Linux用全部入り	eclipse342_rtmttools100release_linux_en.tar.gz MD5:4B1F4ACEE7F8E99B9C36D08068DBE5E1	2010.06.01
Eclipse3.4.2+RTSE+RTCB MacOSX用全部入り	eclipse342_rtmttools100release_macosx_ja.tar.gz MD5:19277C8E1E672688347C6767B57D9D1F	2010.06.10

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 within the Eclipse SDK. The main window shows a system diagram with several components: MyServiceProvider0 (a small green rectangle), ConfigSample0 (a blue rectangle with a dashed border), SequenceOutComponent0 (a large green vertical bar with purple triangles on its right side), and SequenceInComponent0 (a large green vertical bar with purple triangles on its left side). Two horizontal lines connect the right side of SequenceOutComponent0 to the left side of SequenceInComponent0. The left sidebar shows a repository view with a tree structure under 'localhost.localdomain/host_ext', including components like ConfigSample0|rtc, ConsoleIn0|rtc, ConsoleOut0|rtc, manager|mgr, MyServiceProvider0|rtc, SequenceInComponent0|rtc, and SequenceOutComponent0|rtc. The bottom panel shows the configuration view for 'ConfigSample0', with a table of parameters and their values.

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

画面構成

The screenshot shows the RT System Editor interface. The main window displays a System Diagram with components like MyServiceProvider0, ConfigSample0, and SequenceOutComponent0. A Name Service View on the left shows a tree structure of components. A Property View on the right shows the configuration for ConfigSample0.

プロパティ	値
Instance Name	ConfigSample0
Type Name	ConfigSample
Description	Configuration example co...
Version	1.0
Vendor	Moriaki Ando, AIST
Category	example
State	INACTIVE
ExecutionContext	
State	RUNNING
Kind	PERIODIC
Rate	10000

ネームサービスビュー

プロパティビュー

コンフィギュレーションビュー

The Manager Control View shows options for managing modules: Loadable Modules, Loaded Modules, Active Components, Create, Fork, and Shutdown.

マネージャビュー

The Composite Component View shows details for a component named 'ccc' of type 'PeriodicECShared'. It lists components like ConsoleIn0 and ConsoleOut0 with their respective ports.

コンジットコンポーネントビュー

The Execution Context View shows details for an execution context named 'participate0' with a rate of 1000.2. It lists parameters like Name, id, kind, and state.

実行コンテキストビュー

The Log View shows a list of log messages with columns for component, time, level, component, logger, and message. The messages are filtered by level 'ERROR'.

ログビュー

■ Naming Serviceの起動

■ [スタート]メニューから

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[tools]→[Start Naming Service]

■ CosoleInComp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]

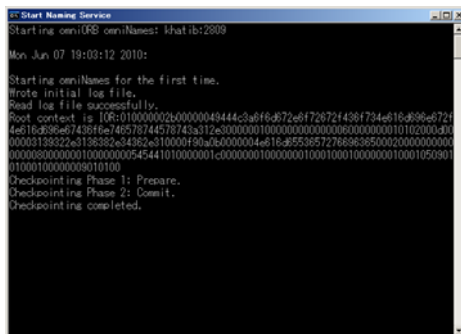
→[examples]→ [ConsoleInComp.exe]

CosoleOutCompの起動

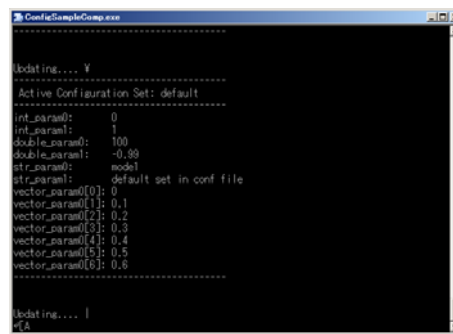
[スタート]メニューから起動

[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]

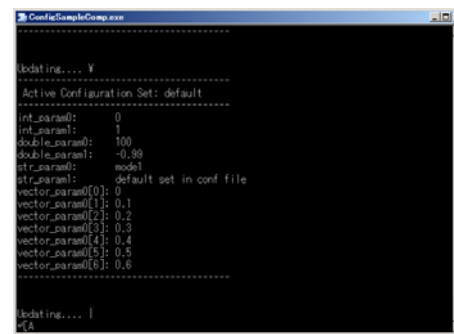
→[examples]→ [ConsoleOutComp.exe]



```
Start Naming Service
Starting omniORB omniNames: khatib:2309
Mon Jun 07 19:03:12 2010:
Starting omniNames for the first time.
Wrote initial ice files.
Read ice file successfully.
Root context is [OR:01000002b00000049444c3a6f6472e6f726721436f734e616d696e672f
4e6168696e67436f6e746578744578743a312e300000100000000000000000000010102000d0
000013822e313822e34382e310000f90a7b5000004e816a955857276698385000200000000
00000000001000000054544101000001c000000100000010001000000001000105090
01000100000009010100]
Checkpointing Phase 1: Prepare.
Checkpointing Phase 2: Commit.
Checkpointing completed.
```



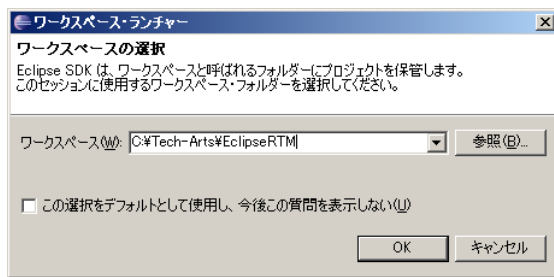
```
ConfigSampleComp.exe
Updating... Y
-----
Active Configuration Set: default
-----
int_param0: 0
int_param1: 1
double_param0: 100
double_param1: -0.89
str_param0: model
str_param1: default set in conf file
vector_param0[0]: 0
vector_param0[1]: 0.1
vector_param0[2]: 0.2
vector_param0[3]: 0.3
vector_param0[4]: 0.4
vector_param0[5]: 0.5
vector_param0[6]: 0.6
-----
Updating... |
%A
```



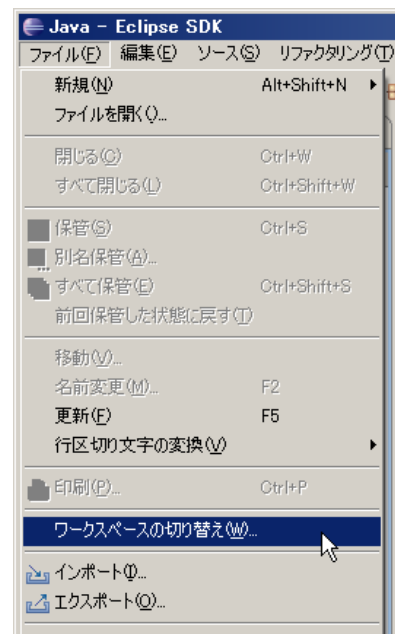
```
ConfigSampleComp.exe
Updating... Y
-----
Active Configuration Set: default
-----
int_param0: 0
int_param1: 1
double_param0: 100
double_param1: -0.89
str_param0: model
str_param1: default set in conf file
vector_param0[0]: 0
vector_param0[1]: 0.1
vector_param0[2]: 0.2
vector_param0[3]: 0.3
vector_param0[4]: 0.4
vector_param0[5]: 0.5
vector_param0[6]: 0.6
-----
Updating... |
%A
```

- Windowsの場合
 - Eclipse.exeをダブルクリック
- Unix系の場合
 - ターミナルを利用してコマンドラインから起動
 - Ex) \$ /usr/local/Eclipse/eclipse

■ ワークスペースの選択(初回起動時)



■ ワークスペースの切替(通常時)



※ワークスペース

Eclipseで開発を行う際の作業領域

Eclipse上でプロジェクトやファイルを作成するとワークスペースとして指定したディレクトリ以下に実際のディレクトリ, ファイルを作成する

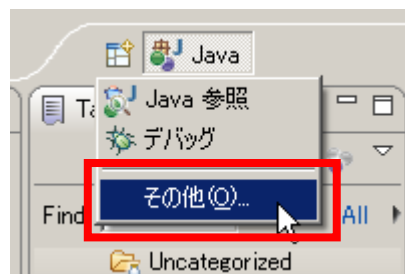
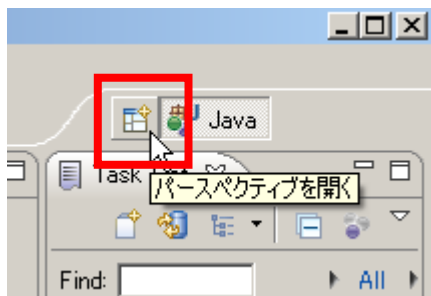
- 初期画面のクローズ
 - 初回起動時のみ



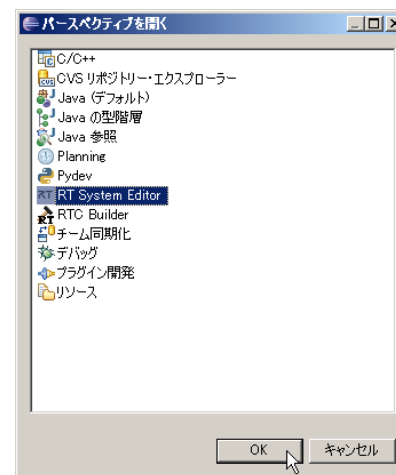
※パースペクティブ
Eclipse上でツールの構成を管理する単位
メニュー, ツールバー, エディタ, ビューなど
使用目的に応じて組み合わせる
独自の構成を登録することも可能

■ 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

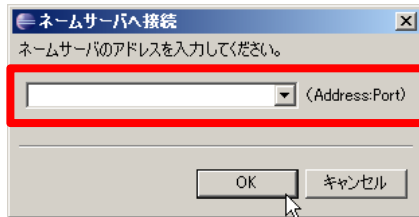
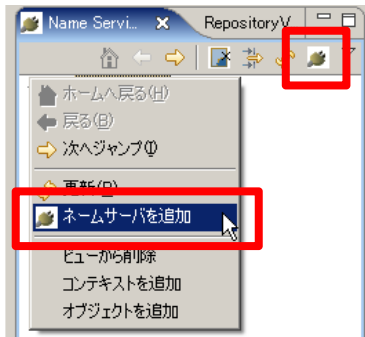


②一覧画面から対象ツールを選択

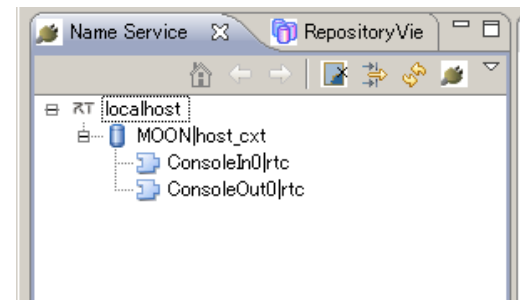


RTシステム構築の基本操作

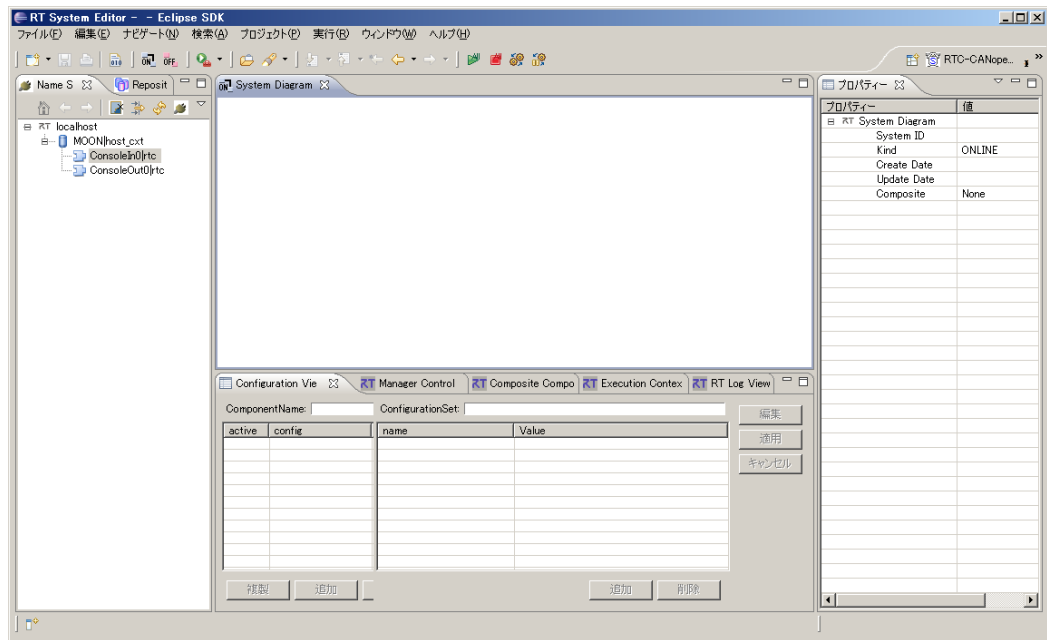
■ ネームサービスへ接続



※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

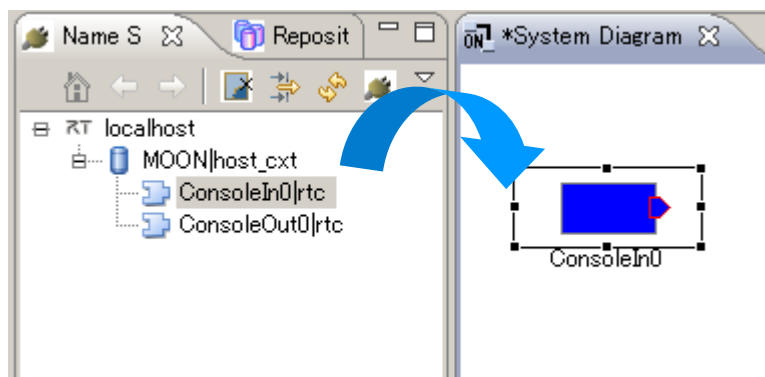


■ システムエディタの起動

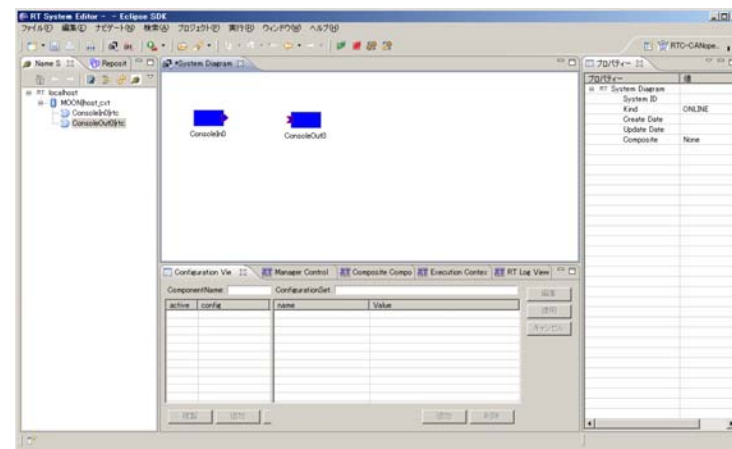


RTシステム構築の基本操作

■ RTコンポーネントの配置

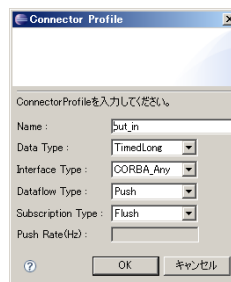
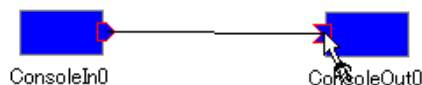


※ネームサービスビューから対象コンポーネントをドラッグアンドドロップ

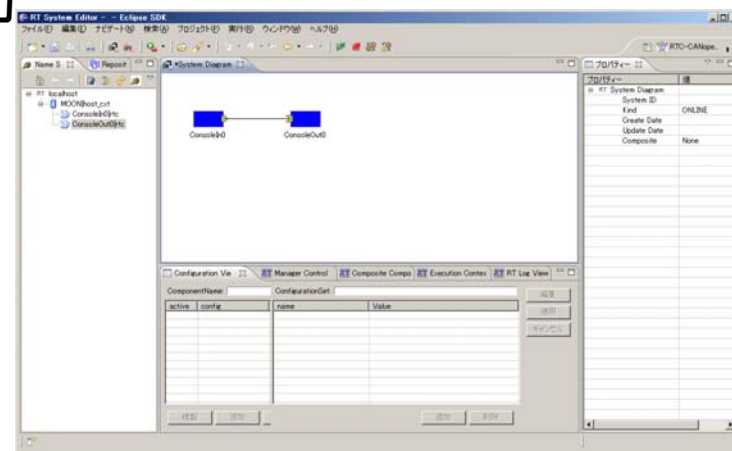


■ ポートの接続

①接続元のポートから接続先の②接続プロフィールを入力
ポートまでドラッグ



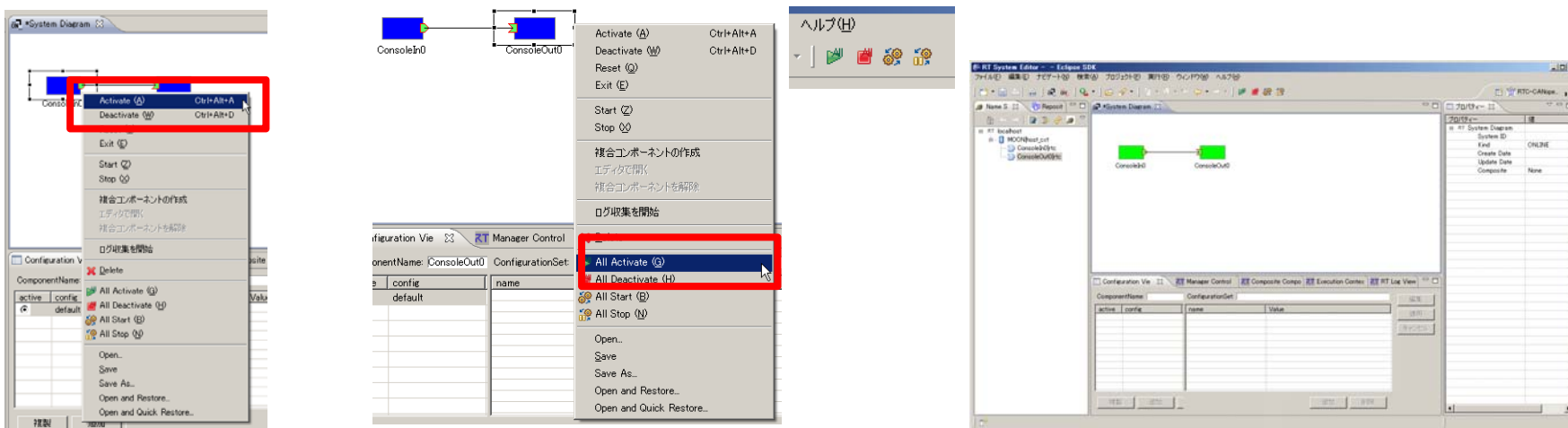
※ポートのプロパティが異なる場合など、
接続不可能なポートの場合にはアイコンが変化



RTシステム構築の基本操作

■ 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



■ サンプルの実行

① ConsoleIn側で数字を入力

② ConsoleOut側が表示

```
ConsoleInComp.exe
dataport.interface_type: corba_cdr
-----
Connector Listener: ON_CONNECT
Profile: name: ConsoleIn0_out_ConsoleOut0.in
Profile: id: 2e25e2b7-cb49-43e1-8b0a-634f0a8dfc75
Profile: properties:
- data_type: IDL:RTC/TimeLong:1.0
- interface_type: corba_cdr
- dataflow_type: push
- subscription_type: flush
- publisher
- push_policy: all
- serializer
- cdr
- endian: little.big
- corba_cdr
- inport_iid: IOR:010000001a00000049444c3a4f70856e52544d2f498e506f72744364723a
812e3000000010000000000006400000001010200004000003139322e3136382e31312e390000
e0b09e00000f3e3eb194e000152e0000000040000200000000000000000000000000000000054
544101000001c00000010000000100010001000000010001050901010001000000009010100
- consumer:
Please input number: 128
```

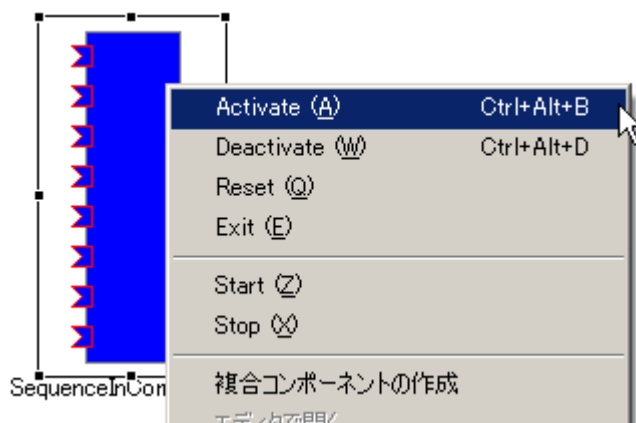
```
ConsoleOutComp.exe
- dataflow_type: push
- subscription_type: flush
- publisher
- push_policy: all
- serializer
- cdr
- endian: little.big
- provider:
-----
Data Listener: ON_RECEIVED
Profile: name: ConsoleIn0_out_ConsoleOut0.in
Profile: id: 2e25e2b7-cb49-43e1-8b0a-634f0a8dfc75
Data: 128
-----
Data Listener: ON_BUFFER_WRITE
Profile: name: ConsoleIn0_out_ConsoleOut0.in
Profile: id: 2e25e2b7-cb49-43e1-8b0a-634f0a8dfc75
Data: 128
Received: 128
TimeStamp: 0[s] 0[ns]
```

※停止はDeactivateを実行

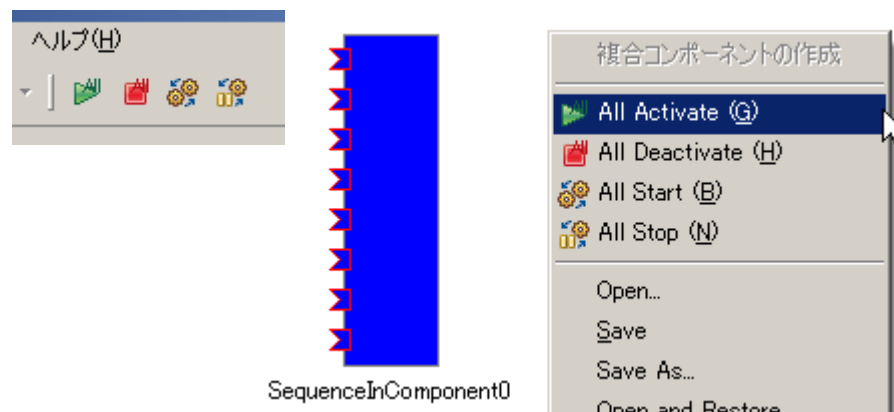
※RTC間の接続を切る場合には
接続線をDelete
もしくは、右クリックメニューから「Delete」を選択

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一定義を追加

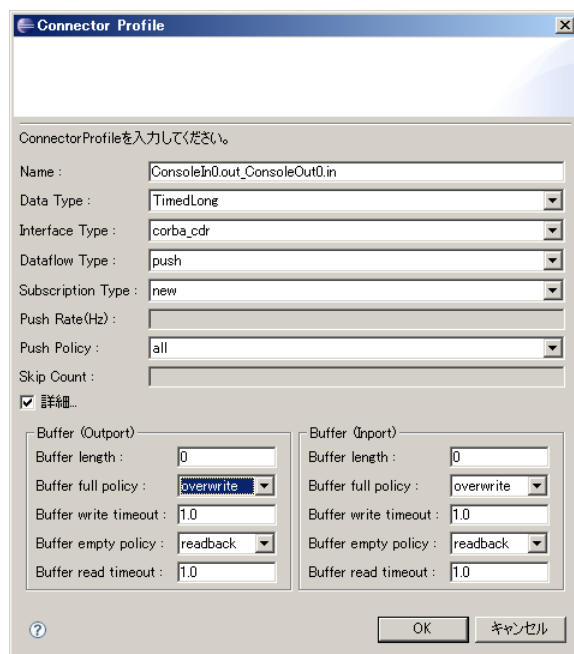
接続プロファイル(DataPort)について

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

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

接続プロファイル(DataPort)について

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



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

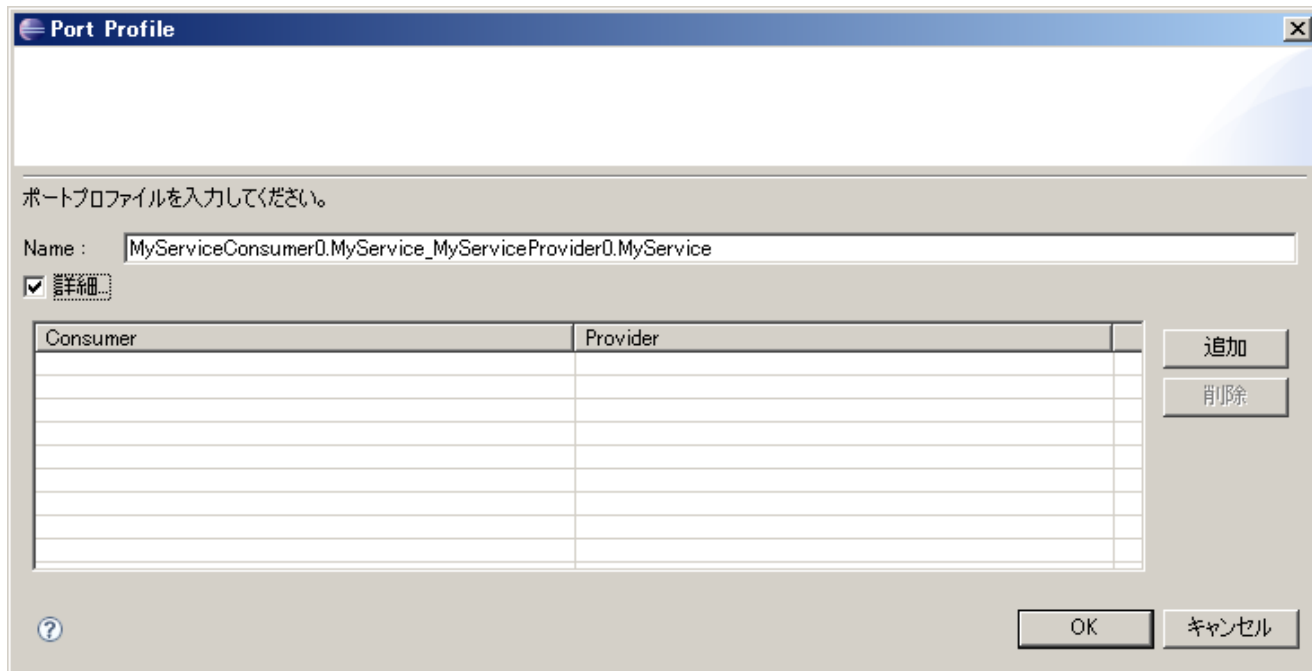
■ Buffer Policy

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

- ※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



ポートプロファイルを入力してください。

Name : MyServiceConsumer0.MyService_MyServiceProvider0.MyService

詳細

Consumer	Provider

追加
削除

OK キャンセル

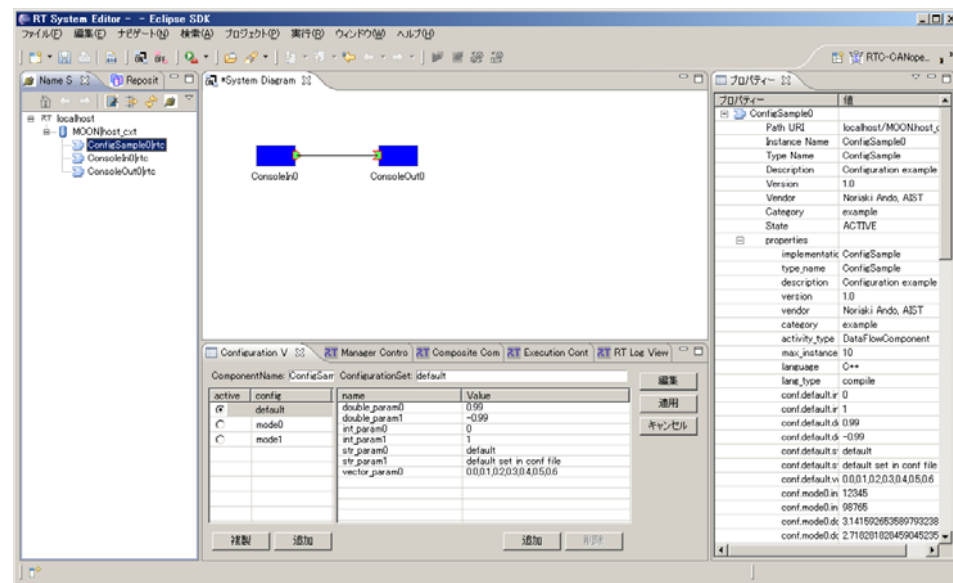
■ ConfigSampleCompの起動

■ [スタート]メニューから起動

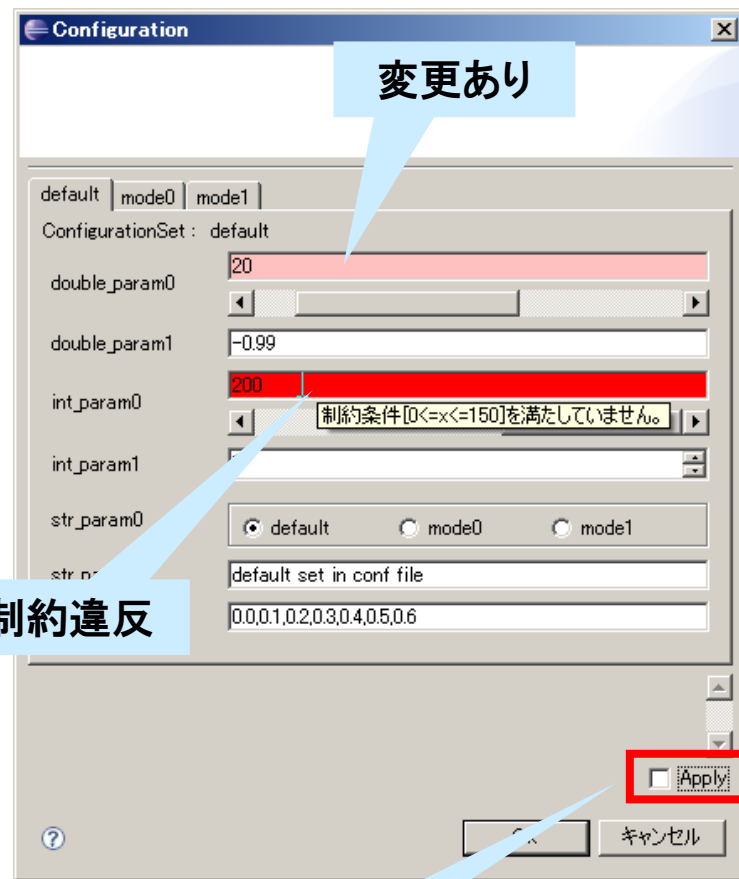
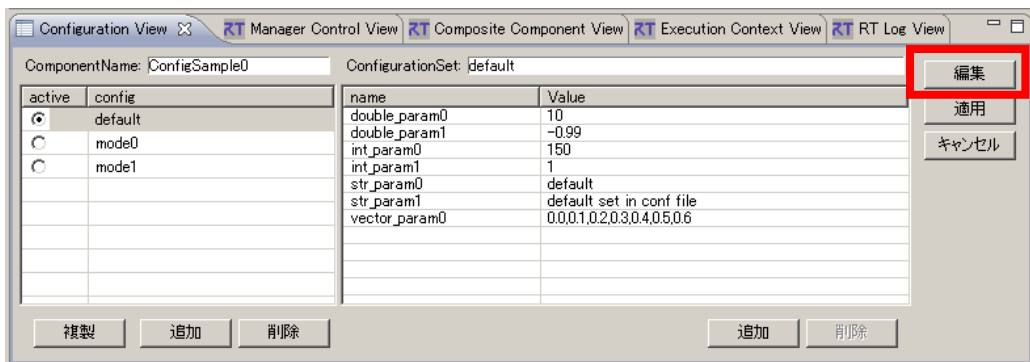
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]

→[examples]→ [ConfigSampleComp.exe]

```
ConfigSampleComp.exe
-----
Updating... ¥
-----
Active Configuration Set: default
-----
int_param0:      0
int_param1:      1
double_param0:   100
double_param1:   -0.99
str_param0:      model
str_param1:      default set in conf file
vector_param0[0]: 0
vector_param0[1]: 0.1
vector_param0[2]: 0.2
vector_param0[3]: 0.3
vector_param0[4]: 0.4
vector_param0[5]: 0.5
vector_param0[6]: 0.6
-----
Updating... |
e[A]
```



■ 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、スライダーを操作する、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

■ 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]

■ OpenCVCameraComp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]

■ 画像処理用コンポーネントの起動

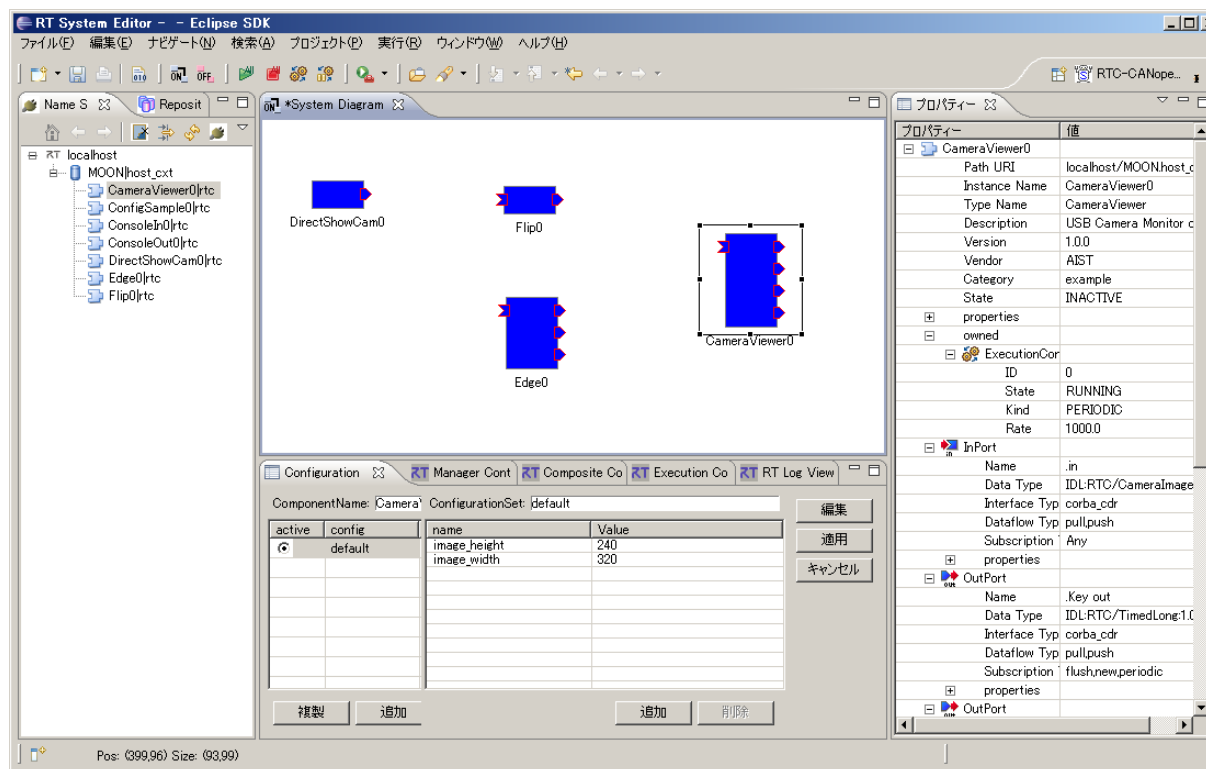
■ [スタート]メニューから起動

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

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

システムの構築

- 以下のコンポーネントをエディタ上に配置
 - DirectShowCam
 - Flip
 - Edge
 - CameraViewer



The screenshot shows the RT System Editor interface. The main window displays a system diagram with four components: DirectShowCam0, Flip0, Edge0, and CameraViewer0. The component tree on the left shows the project structure. The configuration table at the bottom shows the configuration for the selected component (Camera). The properties window on the right shows the configuration for the selected component (CameraViewer0).

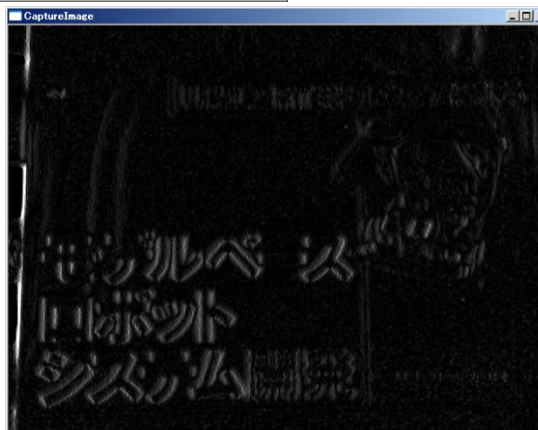
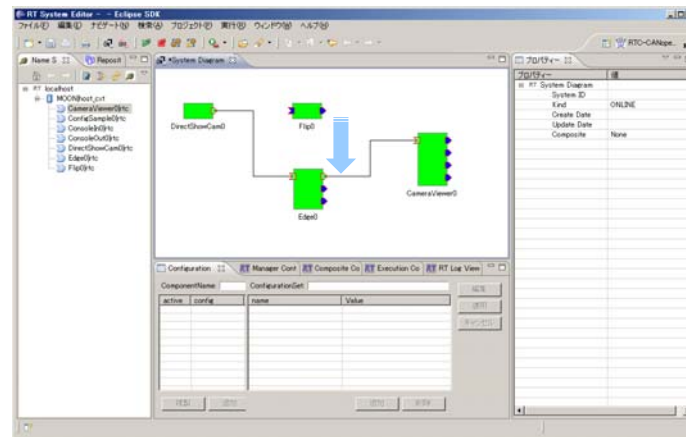
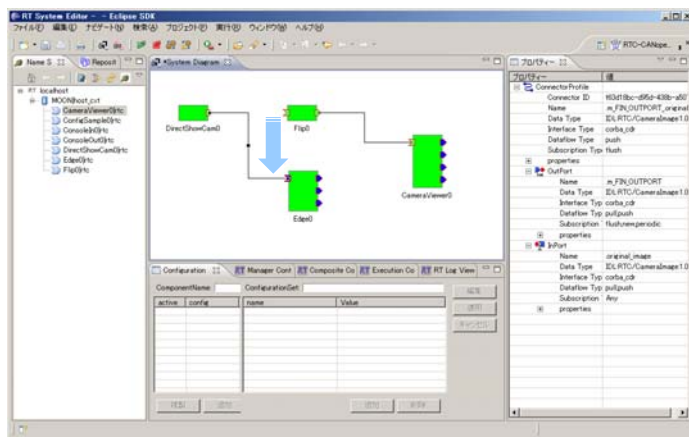
active	config	name	Value
<input checked="" type="checkbox"/>	default	image_height	240
		image_width	320

プロパティ	値
CameraViewer0	
Path URI	localhost/MOON/host_c
Instance Name	CameraViewer0
Type Name	CameraViewer
Description	USB Camera Monitor c
Version	1.0.0
Vendor	AIST
Category	example
State	INACTIVE
properties	
owned	
ExecutionCor	
ID	0
State	RUNNING
Kind	PERIODIC
Rate	1000.0
InPort	
Name	in
Data Type	IDL-RTC/CameraImage
Interface Typ	corba_cdr
Dataflow Typ	pullpush
Subscription	Any
properties	
OutPort	
Name	.Key out
Data Type	IDL-RTC/TimedLong.1.0
Interface Typ	corba_cdr
Dataflow Typ	pullpush
Subscription	flushnew.periodic
properties	
OutPort	

システム構成の変更

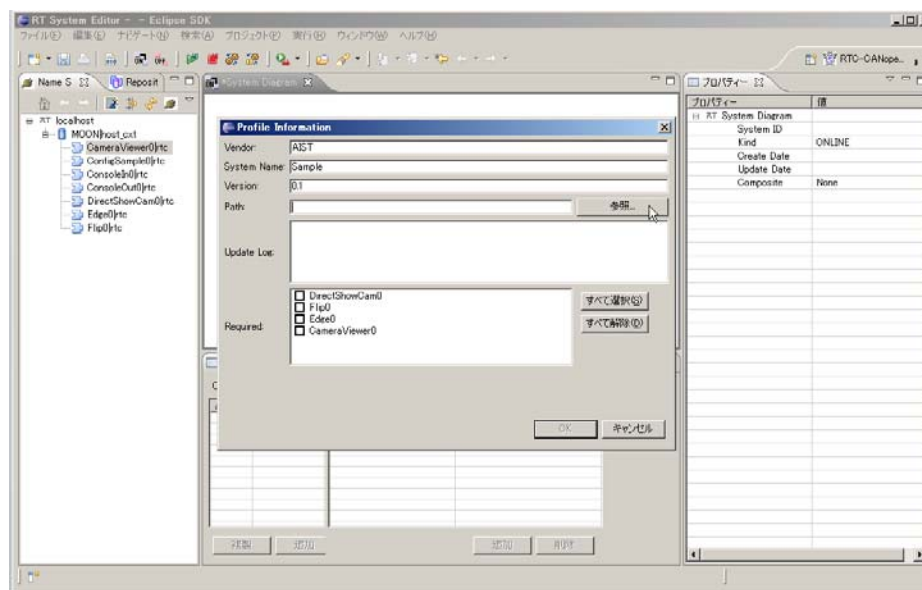
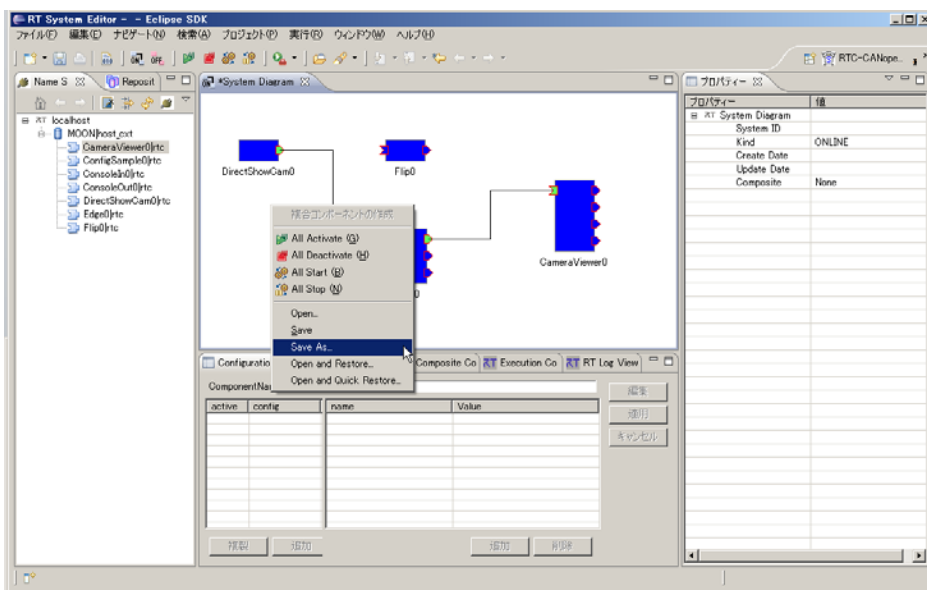
Edge側への差し替え

- 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



システム構成の保存・復元

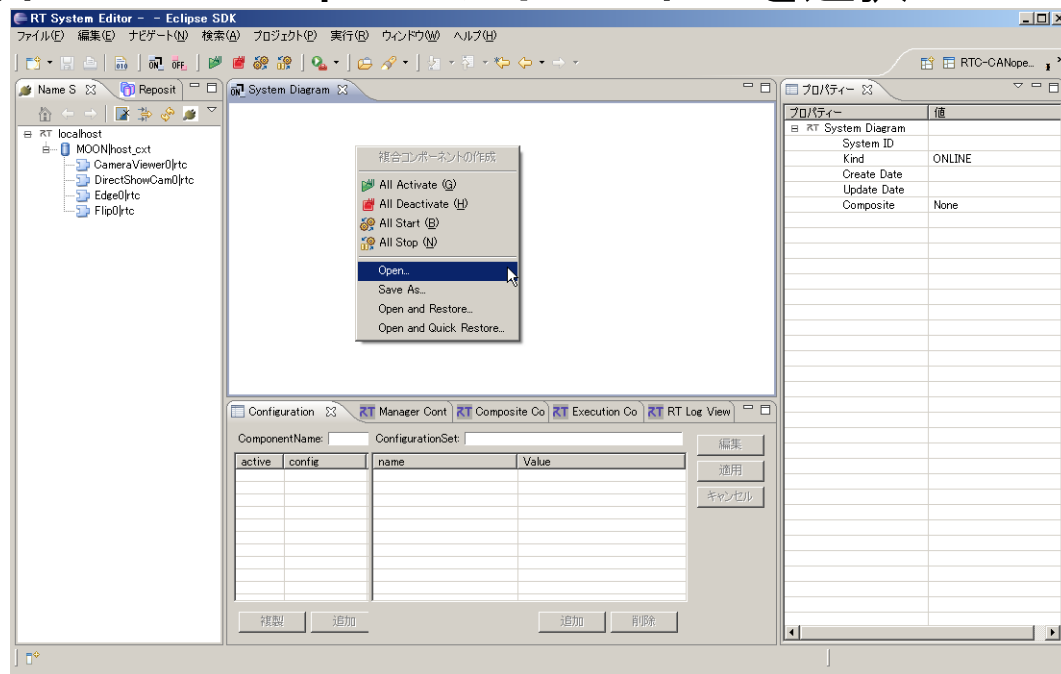
- エディタ上で右クリック
- 表示されたメニュー内から「Save As…」を選択
- 【Profile Information】画面にて、ベンダ名，システム名，バージョン番号，保存先ファイル名を指定



※指定したファイルにXML形式(RtsProfile)で保存

システム構成の保存・復元

- システムエディタを閉じる
- 各コンポーネントを一度終了し, 再起動
- エディタ上で右クリックし, 表示されたメニュー中から以下のどれかを選択
 - 「Open」
 - 「Open and Restore」
 - 「Open and Quick Restore」
- 【ファイル選択画面】にて, 先ほど保存したファイルを指定



※Open: 使用していたRTCのみを読み込み

Open and Restore: 使用していたRTCを読み込むと同時に, 接続, コンフィギュレーションセットの内容も復帰

Open and Quick Restore: 読み込み内容はOpen and Restoreと同様. 該当RTCを検索する際にIORのみを使用

RTCBuilderについて

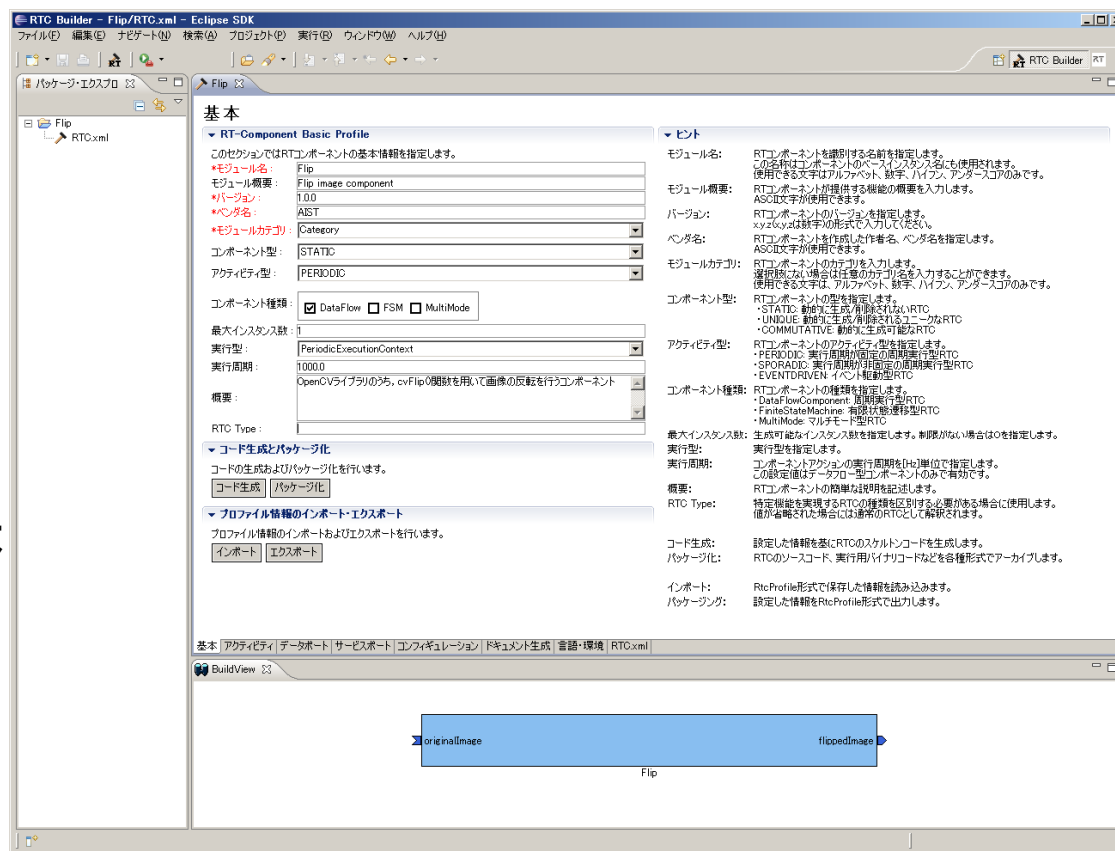


■ RTCBuilderとは？

- 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

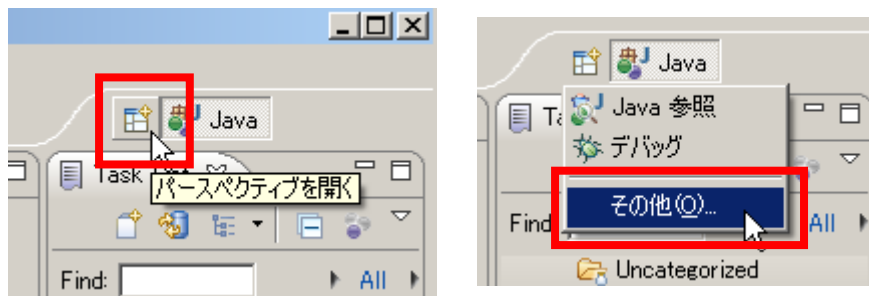
- ※C++用コード生成機能はRtcBuilder本体に含まれています。
- ※その他の言語用コード生成機能は追加プラグインとして提供されています



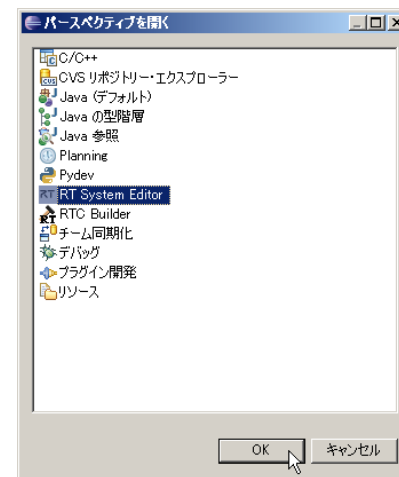
The screenshot shows the RTC Builder application window. The main area is divided into two panes: '基本' (Basic) on the left and 'ヒント' (Hints) on the right. The '基本' pane contains fields for 'RT-Component Basic Profile' such as 'モジュール名' (Module Name), 'バージョン' (Version), 'ベンダ名' (Vendor Name), 'コンポーネント型' (Component Type), and 'コンポーネント種類' (Component Kind). Below these are buttons for 'コード生成とパッケージ化' (Code Generation and Packaging) and 'プロフィール情報のインポート・エクスポート' (Import/Export Profile Information). The 'ヒント' pane provides detailed instructions for each field. A 'パッケージ・エクスプローラ' (Package Explorer) is visible on the left side of the window. At the bottom, the 'BuildView' pane shows a diagram of the 'Flip' component with 'originalImage' and 'flippedImage' ports. Callout boxes highlight these key areas: 'パッケージ・エクスプローラ' (purple), 'RTCプロファイルエディタ' (blue), 'ヒント' (green), and 'ビルドビュー' (pink).

■ 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



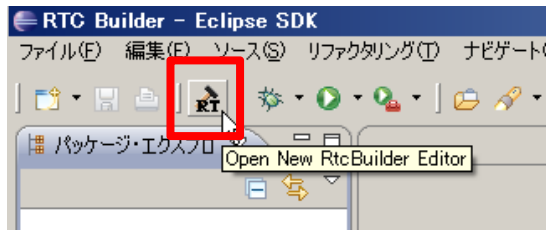
- ② 一覧画面から対象ツールを選択



※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

プロジェクト作成/エディタ起動

① ツールバー内のアイコンをクリック



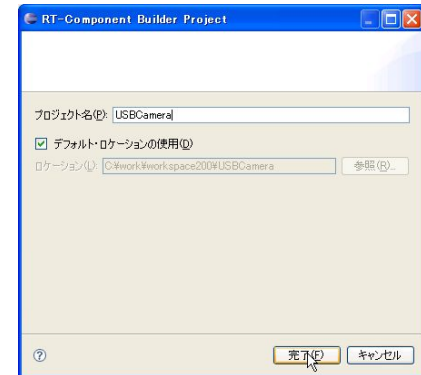
- ※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択し、「次へ」
- ※メニューから「ファイル」-「Open New Builder Editor」を選択

※任意の場所にプロジェクトを作成したい場合

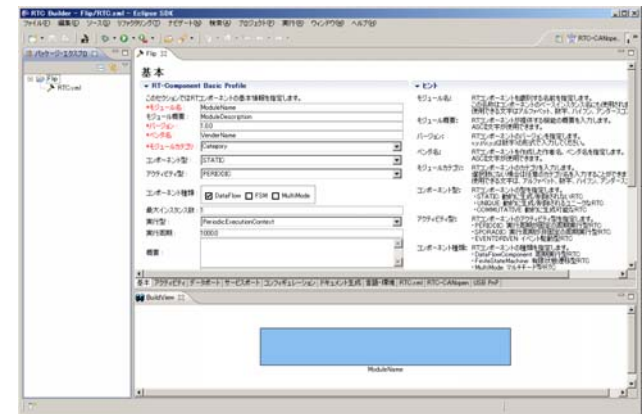
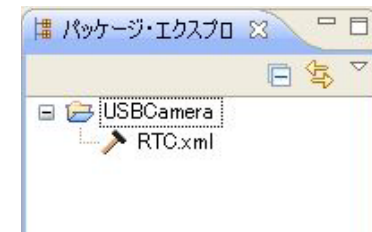
- ②にて「デフォルト・ロケーションの使用」チェックボックスを外す
- 「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名: USBCamera

② 「プロジェクト名」欄に入力し、「終了」

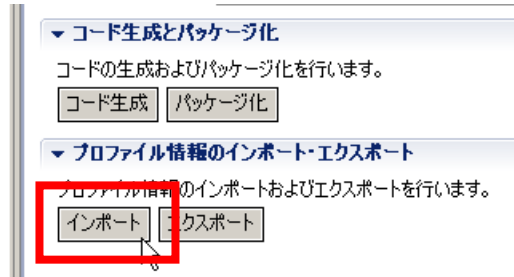


③ 指定した名称のプロジェクトを生成

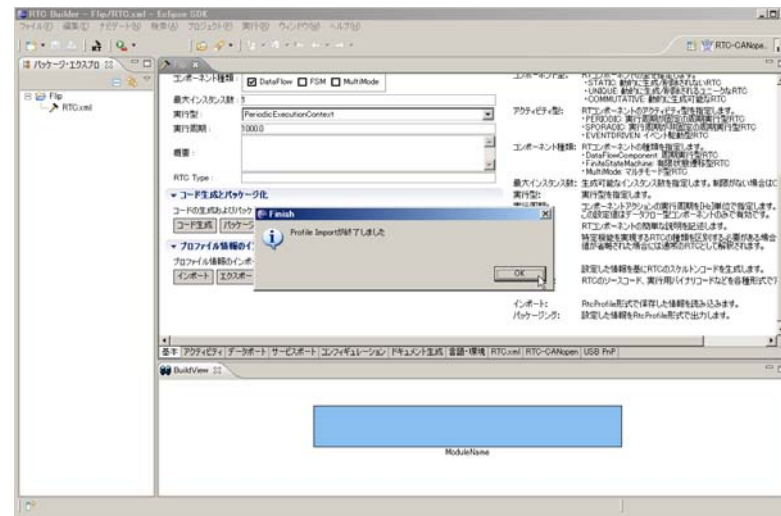


プロファイル インポート

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

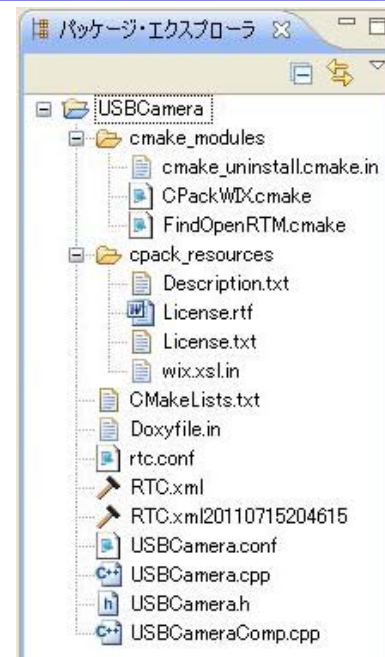
パッケージ化

▼ プロファイル情報のインポート・エクスポート

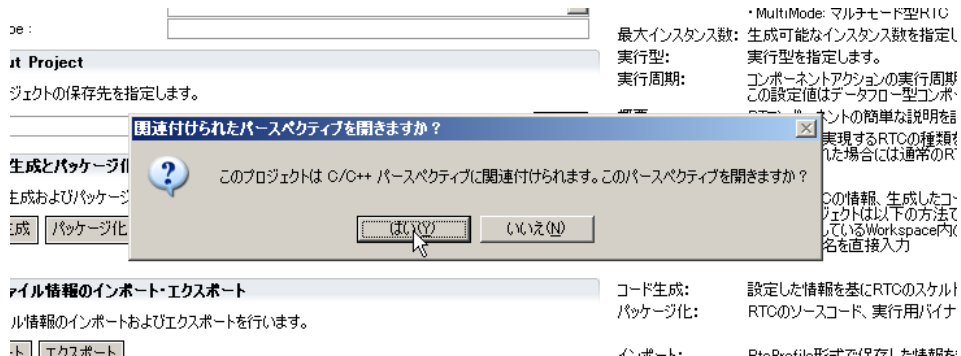
プロファイル情報のインポートおよびエクスポートを行います。

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

画面要素名	説明
基本プロフィール	RTコンポーネントのプロファイル情報など、コンポーネントの基本情報を設定。 コード生成, インポート/エクスポート, パッケージング処理を実行
アクティビティ・プロファイル	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

The screenshot shows the 'RT-Component Basic Profile' configuration window. The 'Basic' tab is active, showing fields for Module Name, Summary, Version, Vendor, Category, Component Type, Activity Type, Component Kind, Max Instances, Execution Type, and Execution Period. The 'Hints' tab is also visible, providing instructions for each field. The 'Code Generation and Packaging' and 'Profile Information' sections are at the bottom.

モジュール名: USB Camera
モジュール概要: 任意(USB Camera 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(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

- ❌ 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

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。

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。

このセクションではヒントを記述します。

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

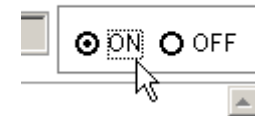
事後条件: コンポーネントの初期化処理が正常に完了している

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

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)	*ポート名 (OutPort)
	image

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: image (OutPort)

*データ型: RTC:CameraImage
変数名: image
表示位置: RIGHT

Documentation

概要説明: Capture images data from the camera

データ型: RTC:CameraImage
データ数:
意味:

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

スポート)の情報を設定します。

② 設定する型情報を一覧から選択

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

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

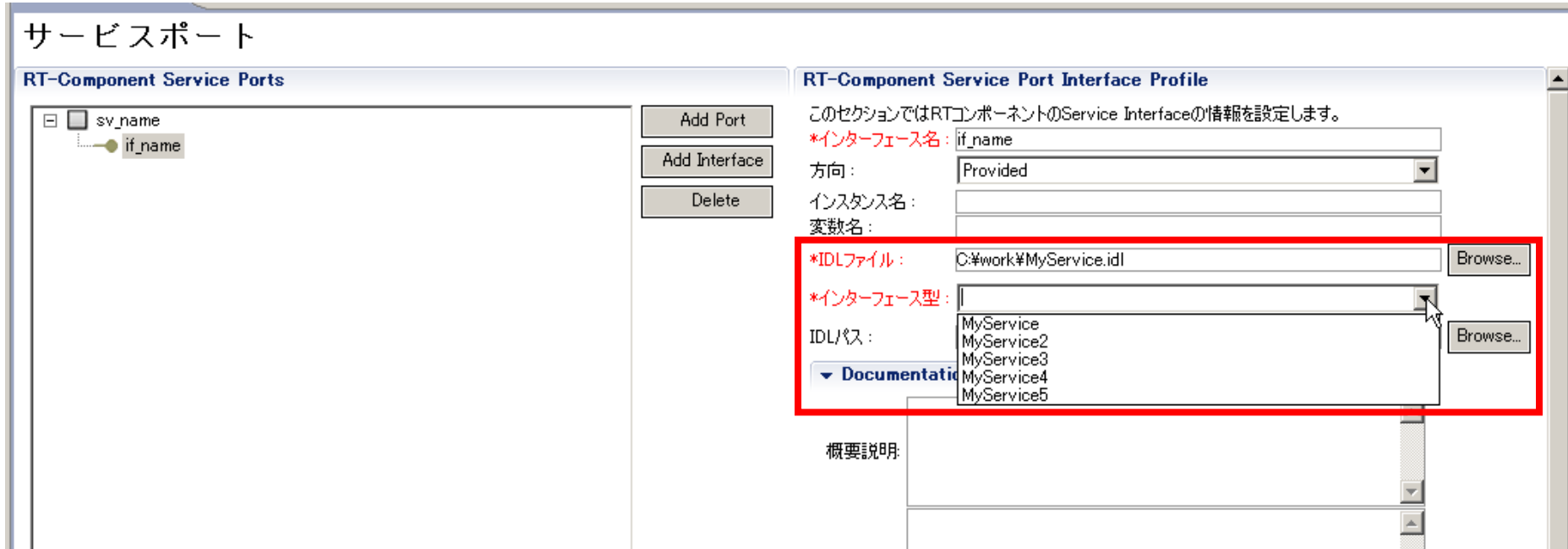
❌ 各ポートに対する説明記述を設定可能
→ 記述した各種コメントは、生成コード内にDoxygen形式で追加される

※Portの設定内容に応じて、下部のBuildViewの表示が変化



- OutPort
ポート名: **image**
データ型: **RTC::CameraImage**
変数名: **image**
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

■ 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

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
deviceNumber

Add Delete

▼ ビット

Config. Param: RTコンポーネントの再利用率をパラメータとして指定します。

パラメータ名: コンフィギュレーション・パラメータ名は名前にはアルファベットと数字のみを使用してください。

データ型: コンフィギュレーション・パラメータの基本型は、int, short, long, float, double, string, enum, array, struct, union, pointer, void, char, wchar_t, bool, complex, user-defined, etc.です。

デフォルト値: コンフィギュレーション・パラメータのデフォルト値を指定します。

変数名: コンフィギュレーション・パラメータの実際の名称を指定します。

単位: コンフィギュレーション・パラメータの単位を指定します。

制約条件: コンフィギュレーション・パラメータの制約条件を指定します。

Widget: コンフィギュレーション・パラメータのWidgetを指定します。

Step: コンフィギュレーション・パラメータのStepを指定します。

▼ 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: deviceNumber

*データ型: int

*デフォルト値: 0

変数名:

単位:

制約条件:

Widget: text

Step:

Documentation:

①「Add」ボタンをクリックし、追加後、直接入力で名称設定

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: **deviceNumber**
データ型: **int**
デフォルト値: **0**
変数名: **deviceNumber**
制約条件:
Widget: **text**

※データ型は、short,int,long,float,double,stringから選択可能(直接入力も可能)

※制約情報とWidget情報を入力することで、RTSystemEditorのコンフィギュレーションビューの表示を設定することが可能

■ 制約条件について

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

■ 制約の記述書式

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

■ Widget

- text(テキストボックス)
 - デフォルト
- slider(スライダ)
 - **数値型**に対して**範囲指定**の場合
 - 刻み幅をstepにて指定可能
- spin(スピナ)
 - **数値型**に対して**範囲指定**の場合
 - 刻み幅をstepにて指定可能
- radio(ラジオボタン)
 - 制約が**列挙型**の場合に指定可能

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

■ 生成対象RTCを実装する言語，動作環境に関する情報を設定

言語・環境

▼ 言語

このセクションでは使用する言語を指定します

- C++
- Python
- Java
- Ruby

Use old build environment.

▼ 環境

このセクションでは依存するライブラリや使用するOSなどを指定します

Version	OS

Add
Delete

詳細情報

OS Version

Add
Delete

CPU

Add
Delete

▼ ヒント

言語: RTコンポーネントを作成する言語を選択します。リスト中の言語から選択可能です。

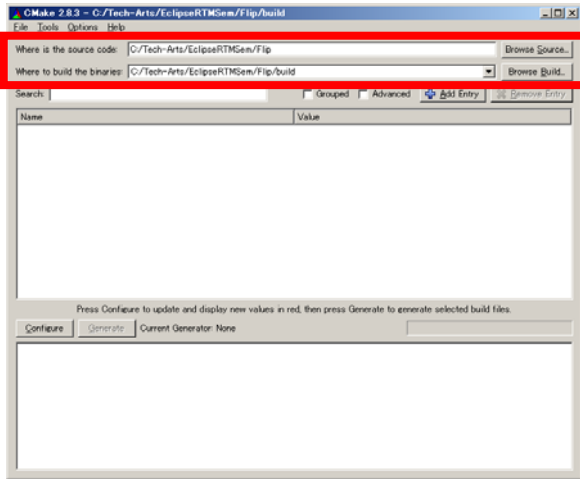
環境: 言語ごとのライブラリの依存関係や、使用するOSなどの環境を選択します。
詳細情報で設定した内容(OS情報、ライブラリ情報など)は、プロファイル内にも保存されます。

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

コンパイル(Windows, CMake利用)

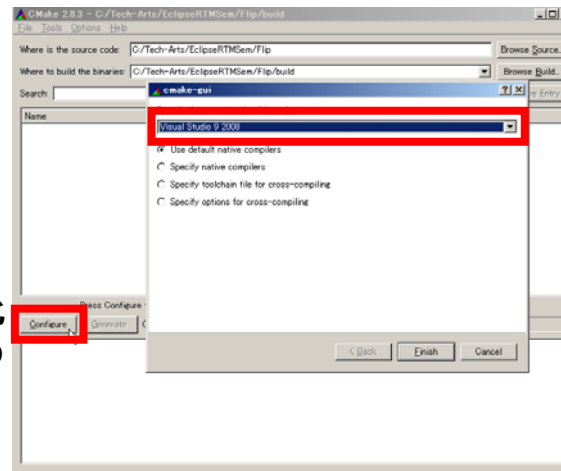
① GUI版Cmakeを起動し, source, binaryのディレクトリを指定



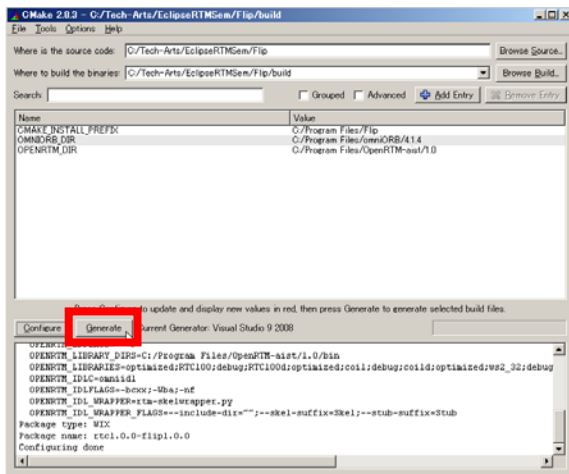
※binaryには, sourceとは別のディレクトリを指定する事を推奨

※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

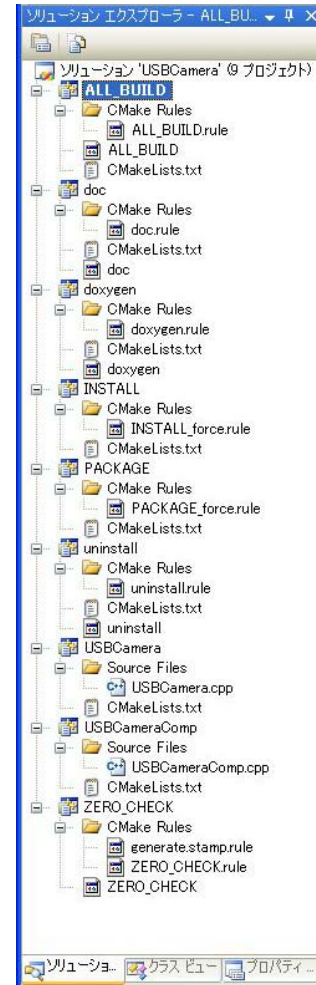
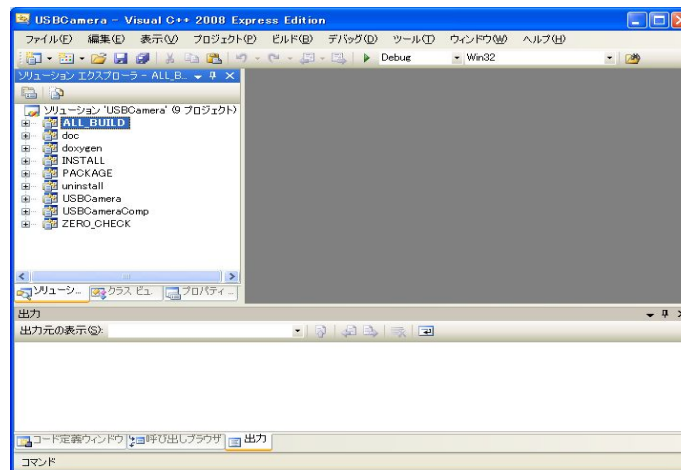
② 「Configure」を実行し, 使用するプラットフォームを選択



③ 正常終了後, 「Generate」を実行

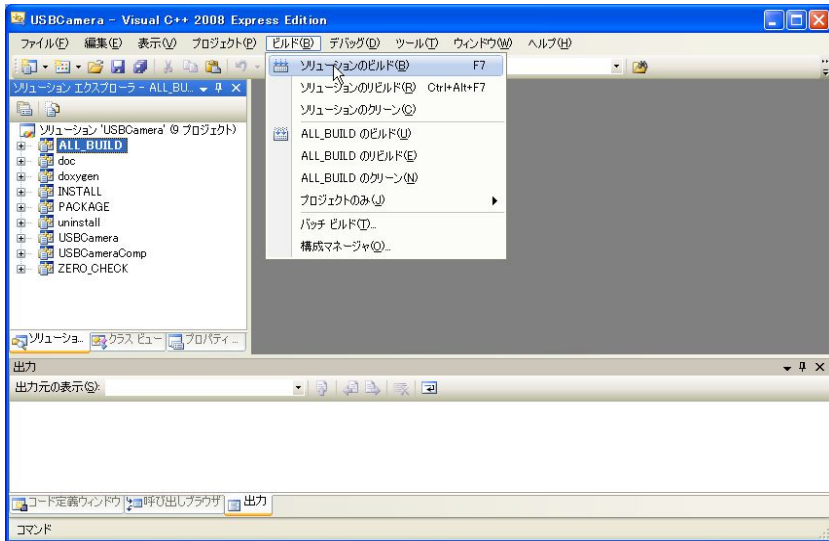


④ binaryとして指定したディレクトリ内にあるソリューションファイルを開く

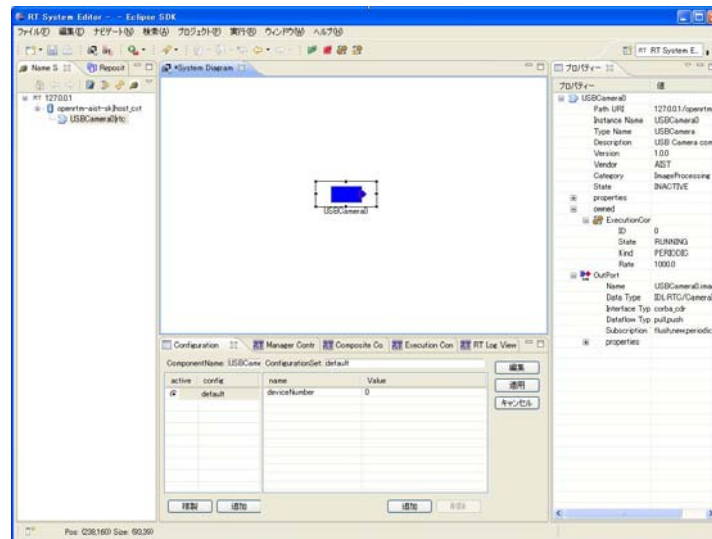


コンパイル・実行(Windows, CMake利用)

⑤ ソリューションをビルド

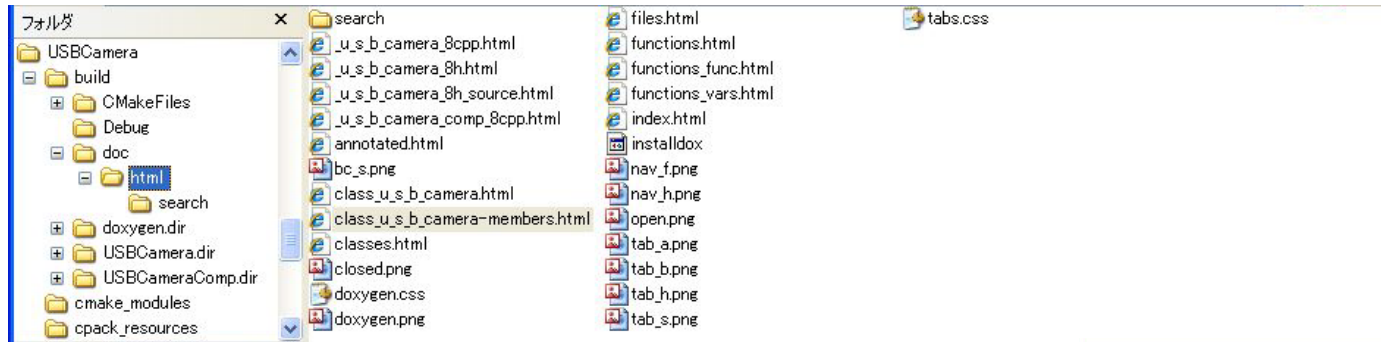


⑥ binaryにて指定したディレクトリ以下のDebug内のUSBcameraComp.exeを起動



ドキュメント作成 (Windows, CMake利用)

※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

usbcamera 1.0.0

メインページ クラス ファイル

検索

クラス **USBCamera**

USB Camera component. [詳細]

```
#include <USBCamera.h>
```

すべてのメンバー一覧

Public メソッド

virtual RTC::ReturnCode_t	onInitialize ()
virtual RTC::ReturnCode_t	onActivated (RTC::UniqueId ec_id)
virtual RTC::ReturnCode_t	onDeactivated (RTC::UniqueId ec_id)
virtual RTC::ReturnCode_t	onExecute (RTC::UniqueId ec_id)

Protected 変数

int	m_deviceNumber
CameraImage	m_m_image
OutPort< CameraImage >	m_m_imageOut

説明

RTC::ReturnCode_t USBCamera::onDeactivated (RTC::UniqueId ec_id) [virtual]

Post-processing such as freeing allocated memory.

RTC::ReturnCode_t USBCamera::onExecute (RTC::UniqueId ec_id) [virtual]

Capture images from the camera, and outputs the data from OutPort.

変数

int USBCamera::m_deviceNumber [protected]

Device number

- Name: deviceNumber deviceNumber
- DefaultValue: 0

OutPort< CameraImage > USBCamera::m_m_imageOut [protected]

Capture images data from the camera

- Type: RTC::CameraImage

このクラスの説明は次のファイルから生成されました:

- C:/work/workspace199/USBcamera/USBcamera.h
- C:/work/workspace199/USBcamera/USBcamera.cpp

usbcamera1に対してFri Jul 15 2011 21:41:19に生成されました. [doxygen](#) 1.7.3

usbcamera 1.0.0

メインページ クラス ファイル

ファイル一覧

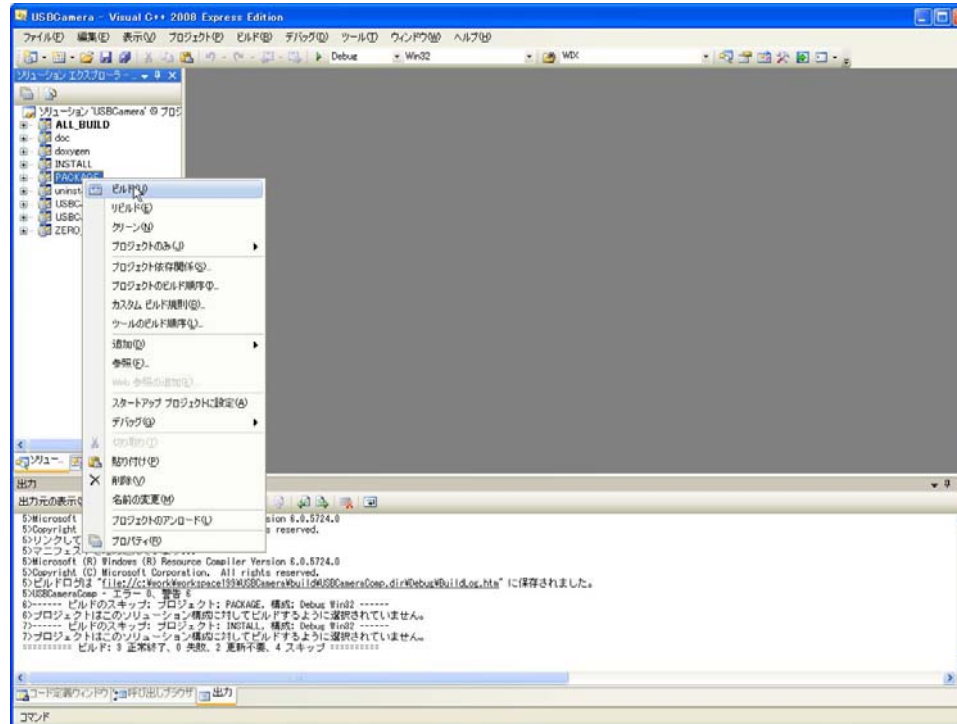
C:/work/workspace199/USBcamera/USBcamera.h

説明を見る。

```
00001 // -*- C++ -*-
00015 #ifndef USBCAMERA_H
00016 #define USBCAMERA_H
00017
00018 #include <rtm/Manager.h>
00019 #include <rtm/DataFlowComponentBase.h>
00020 #include <rtm/CorbaPort.h>
00021 #include <rtm/DataInPort.h>
00022 #include <rtm/DataOutPort.h>
00023 #include <rtm/idl/BasicDataTypesSkel.h>
00024 #include <rtm/idl/ExtendedDataTypesSkel.h>
00025 #include <rtm/idl/InterfaceDataTypesSkel.h>
00026
00027 // Service implementation headers
00028 // <rtc-template block="service_impl_h">
00029
00030 // </rtc-template>
00031
00032 // Service Consumer stub headers
00033 // <rtc-template block="consumer_stub_h">
00034
00035 // </rtc-template>
00036
00037 using namespace RTC;
00038
00039
00040 class USBCamera
00041 : public RTC::DataFlowComponentBase
00042 {
00043 public:
00044     USBCamera (RTC::Manager* manager);
00045     ~USBCamera ();
00046
00047     // <rtc-template block="public_attribute">
```

配布用パッケージ作成(Windows,CMake利用)

■ 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

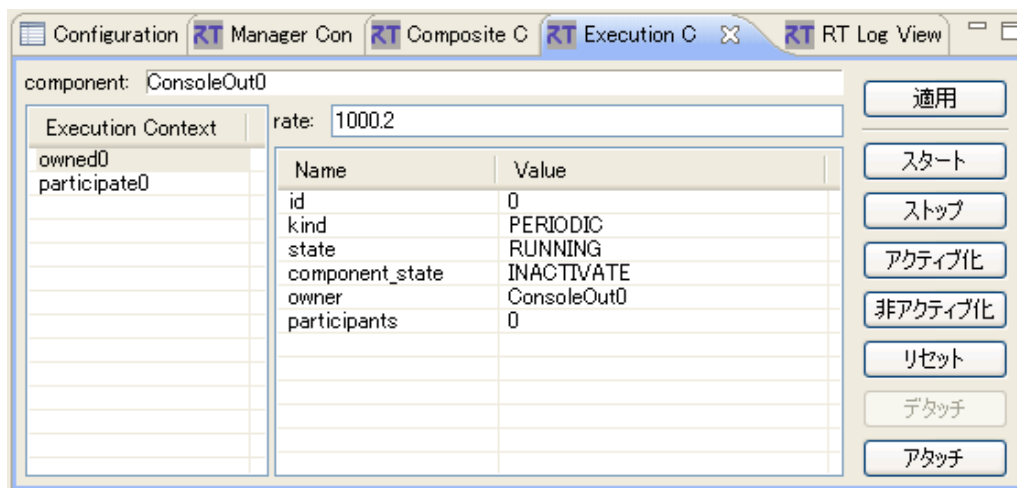
● コンポーネントのインストール先

C:\Program Files\OpenRTM-aist\1.1\components\<言語>\<パッケージ名>

RTSystemEditor 補足説明



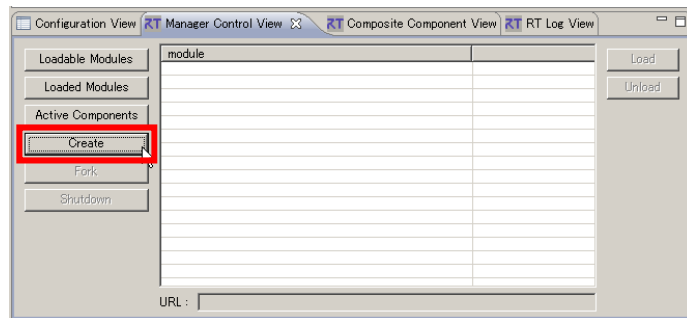
■ RTコンポーネントが属する実行コンテキスト(EC)を一覧表示



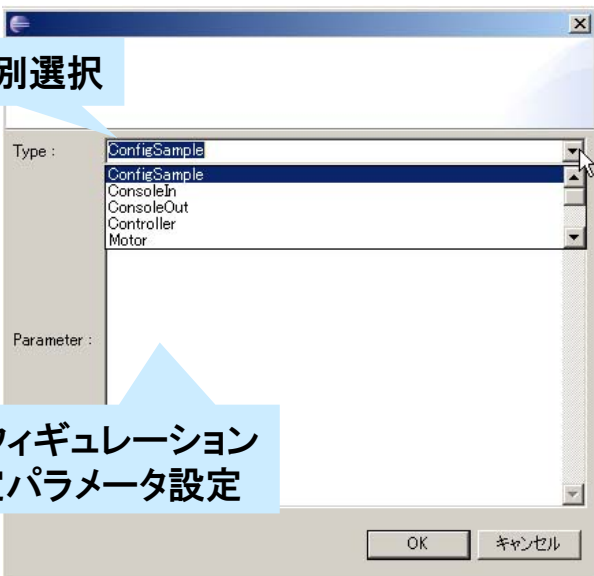
属性名	説明
id	ECのID. オンラインの場合には, context_handleを表示
kind	ECの種別(PERIODIC/EVENT_DRIVEN/OTHER)
state	ECの状態(RUNNING/STOPPING)
component state	対象RTCの状態(ACTIVE/INACTIVE/ERROR)
owner	対象ECを所有しているオーナーRTCのインスタンス名
participants	対象ECに参加中のRTCの数

※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

■ RTコンポーネントの新規インスタンスの生成

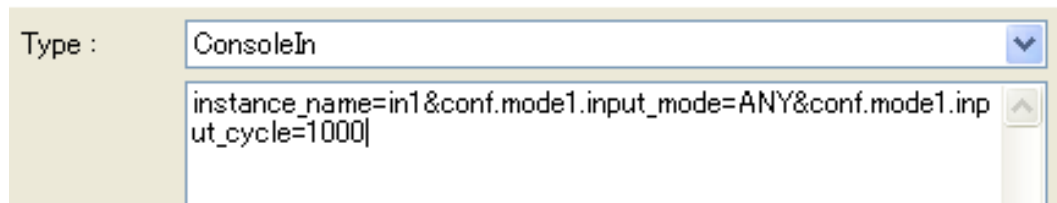


RTC種別選択

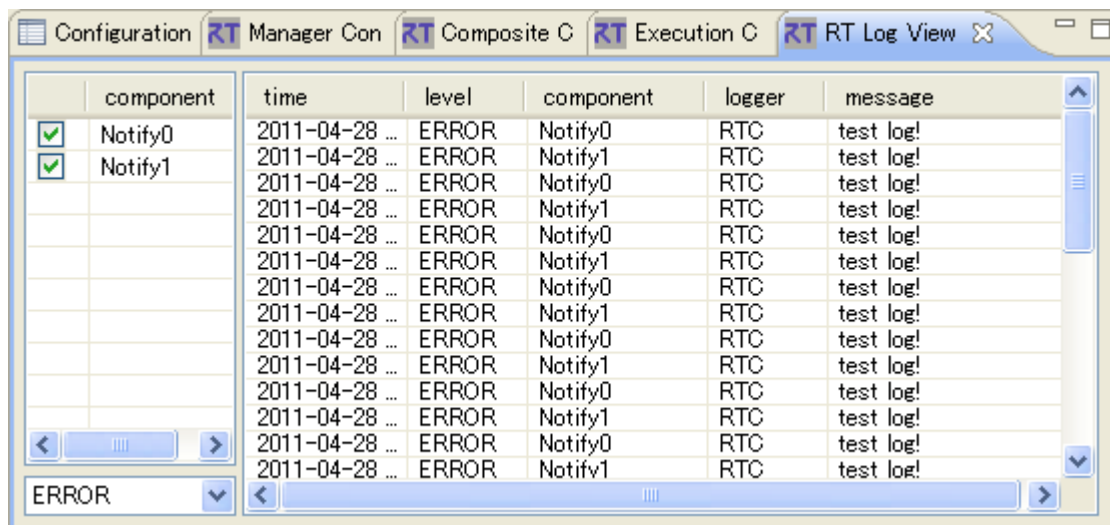


コンフィギュレーション
指定パラメータ設定

- 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

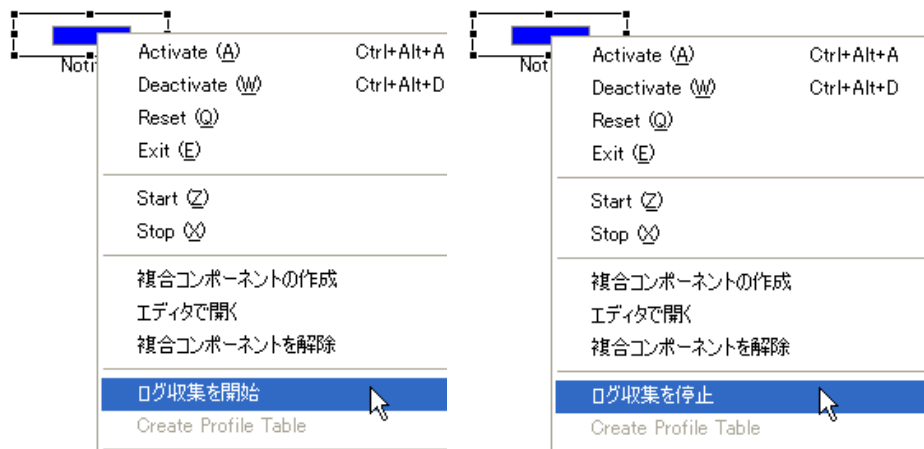


■ 選択したRTCから収集したログ情報を一覧表示

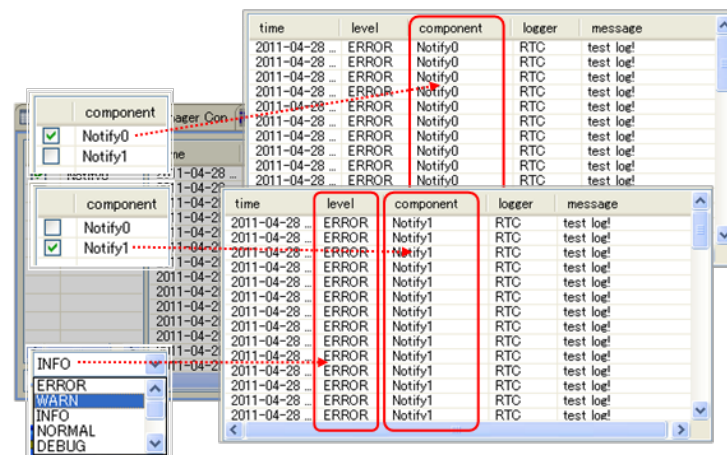


※近日機能追加予定

● ログ収集の開始/停止



● ログ情報のフィルタリング

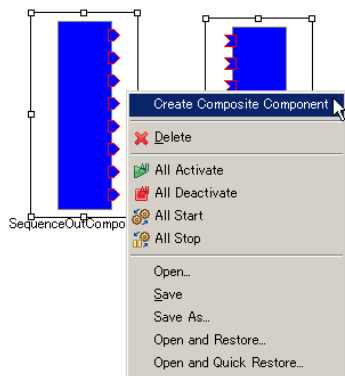


複合コンポーネント

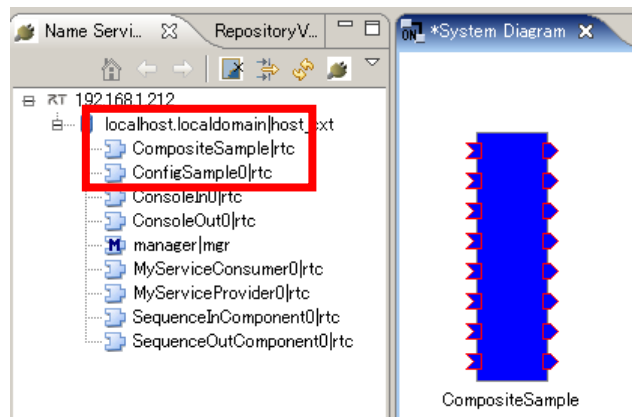
- 複数のRTCをまとめて、1つのRTCとして扱うための仕組み

- 複合コンポーネントの作成方法

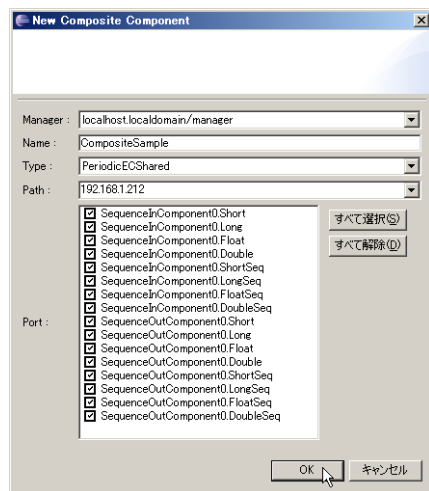
① 複数RTCを選択している状態で右クリック



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

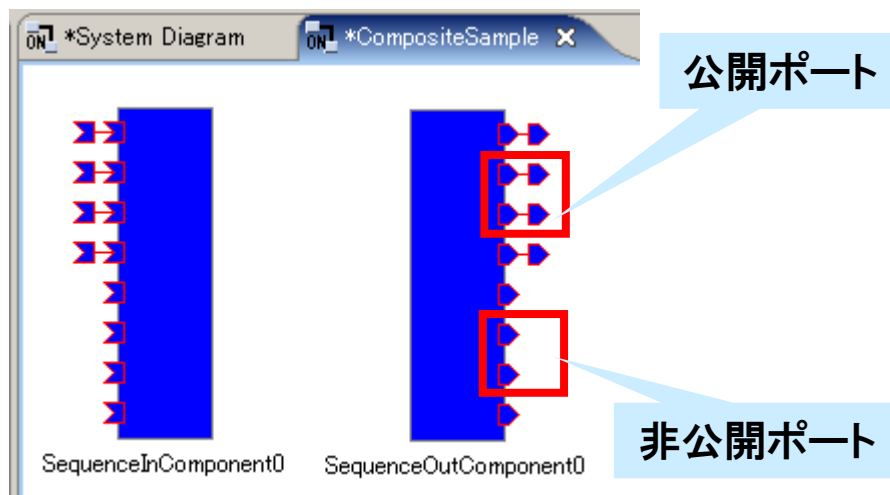
※生成対象複合コンポーネント外部と接続されているPortは強制的に公開されます

■ 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

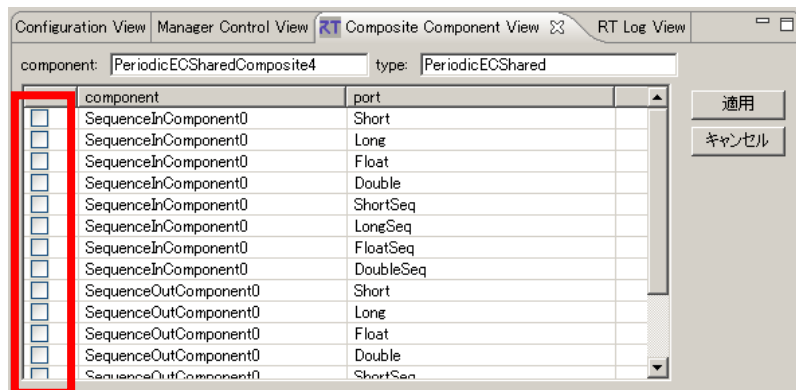


- ※エディタ内に別RTCをDnDすることで、子コンポーネントの追加が可能
→追加したRTCのポートは全て非公開に設定
- ※エディタ内のRTCを削除することで、子コンポーネントの削除が可能
→削除されたRTCは、親エディタに表示

■ 公開ポートの設定

● 複合コンポーネントビュー

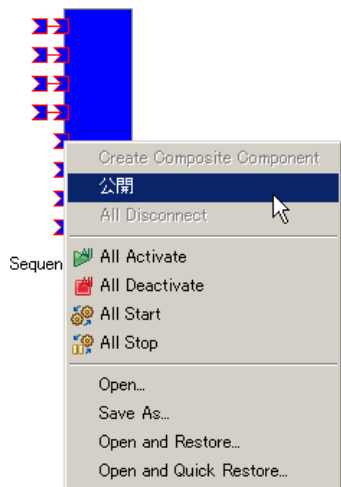
ポート公開情報



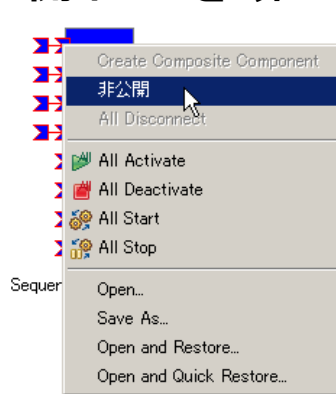
※ポート公開情報を変更し、「適用」をクリック

● 複合コンポーネントエディタ

※非公開ポートを「公開」



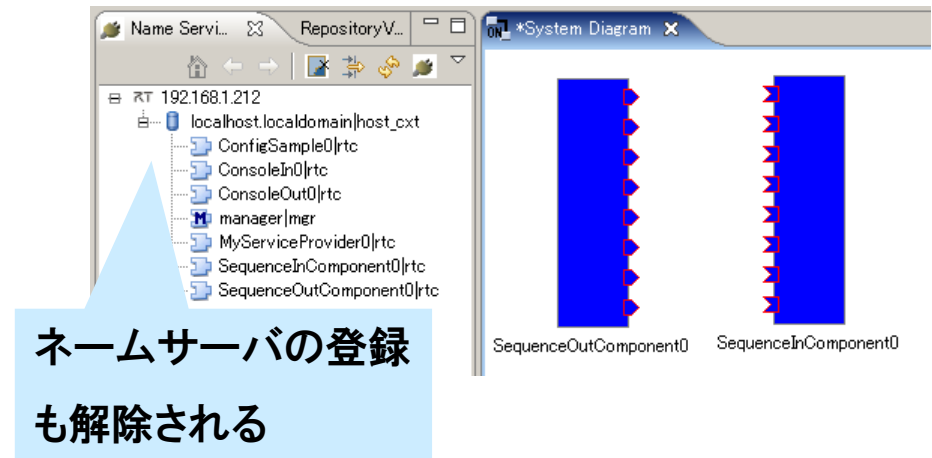
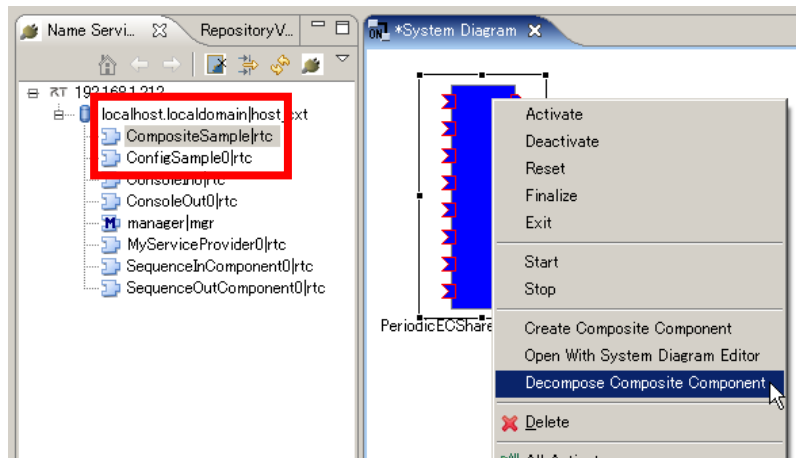
※公開ポートを「非公開」



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



※ 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

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

リポジトリビュー

オフライン・システムエディタ

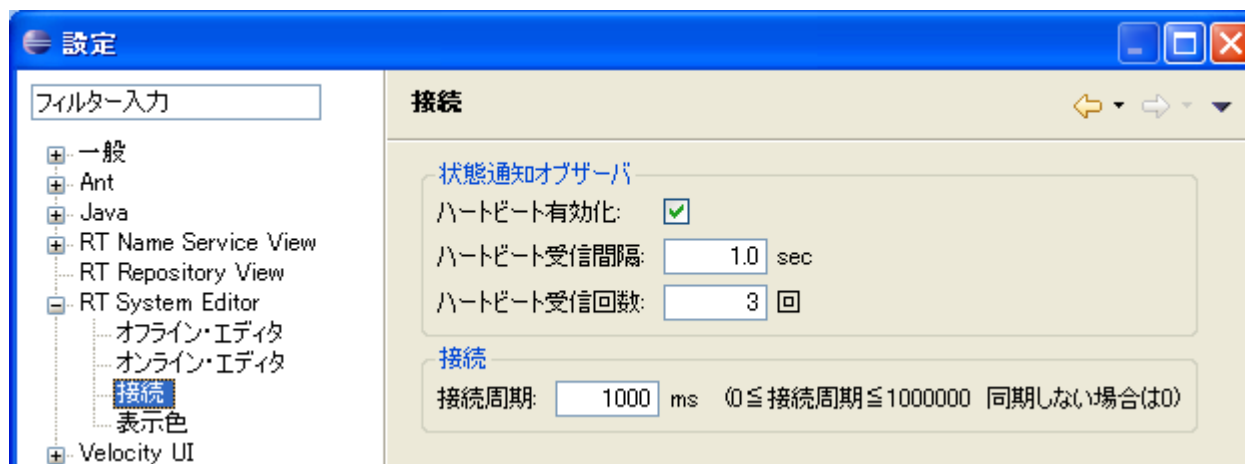
プロパティビュー

Configuration View

ComponentName	ConfigurationSet	active	config	name	Value
ImageProcess_1					

■ 接続 - 状態通知オブザーバ

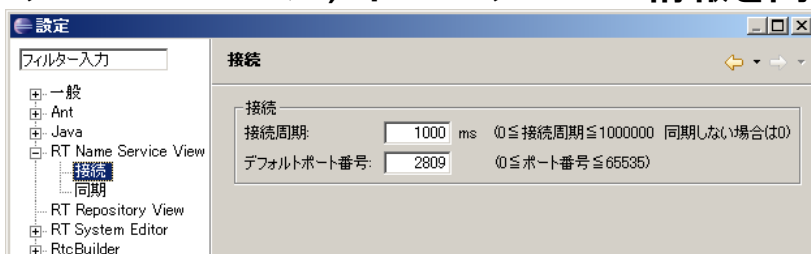
- RTCの生存確認用オブザーバに関する設定
 - RTSE側から生存確認を行うのではなく, RTC側から通知(ハートビート)を行う形
 - OpenRTM-aist-1.1以降で対応



- 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場合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

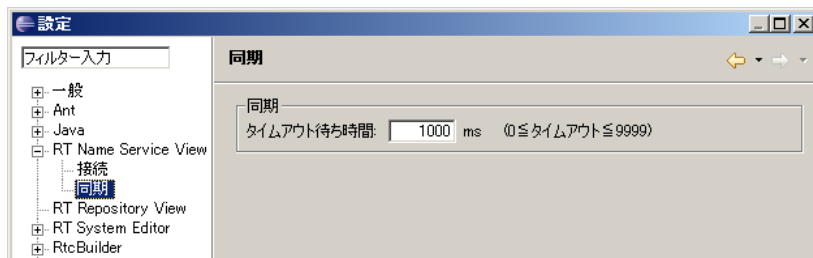
■ 「RT Name Service View」－「接続」【接続周期】

- 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



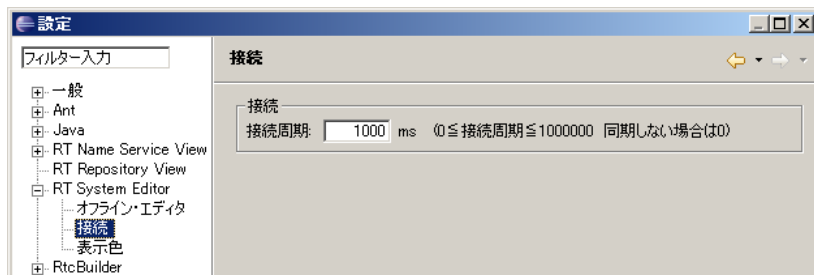
■ 「RT Name Service View」－「同期」【タイムアウト待ち時間】

- 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」－「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



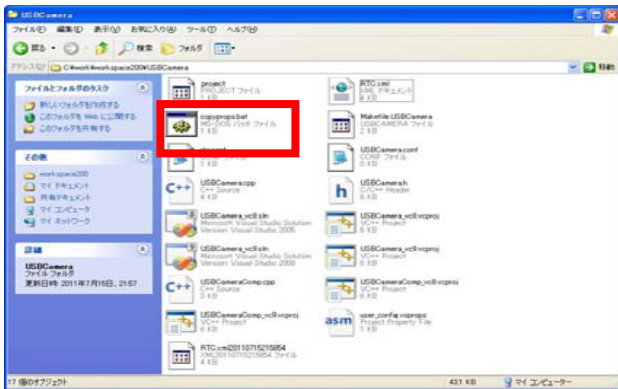
**【接続周期】をゼロに設定すると
ネームサーバとの同期を行わない**

RTCBuilderの補足

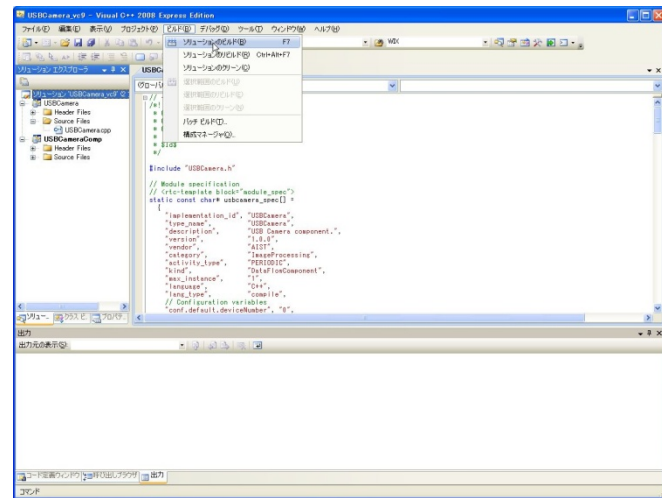


コンパイル・実行(Windows)

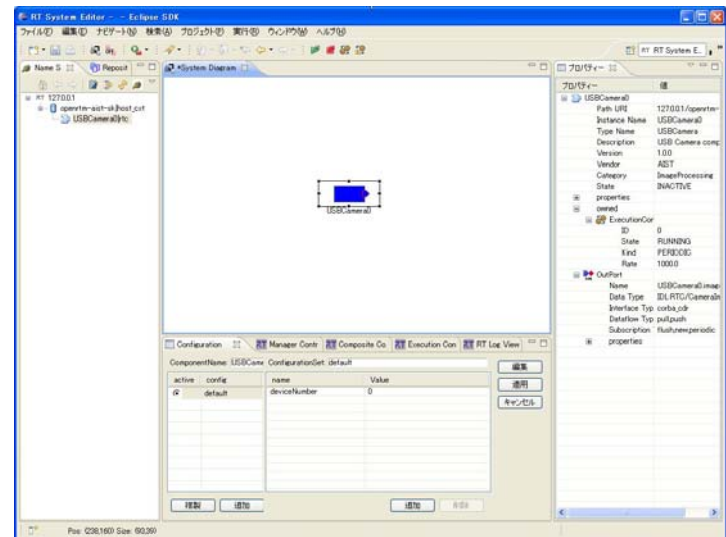
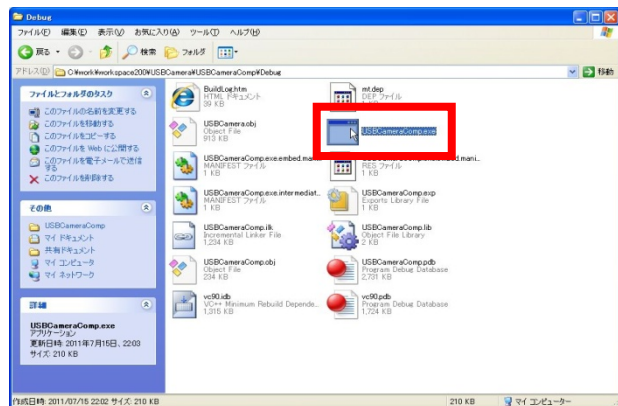
①コード生成先ディレクトリ内の「copyprops.bat」をダブルクリックして、設定ファイルをコピー



②VisualStudioを用いたビルド

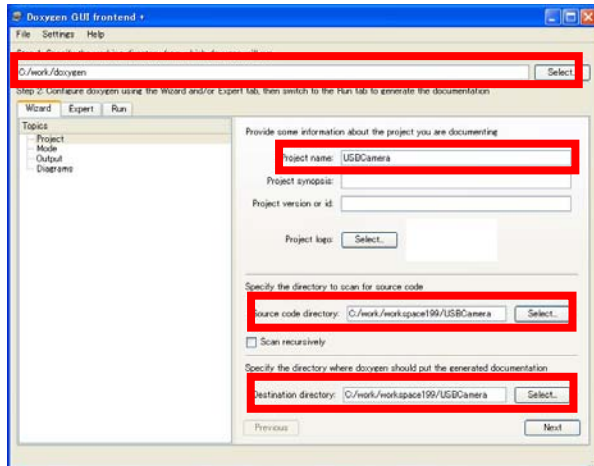


③USBCameraComp¥¥Debug内のUSBCameraComp.exeを起動

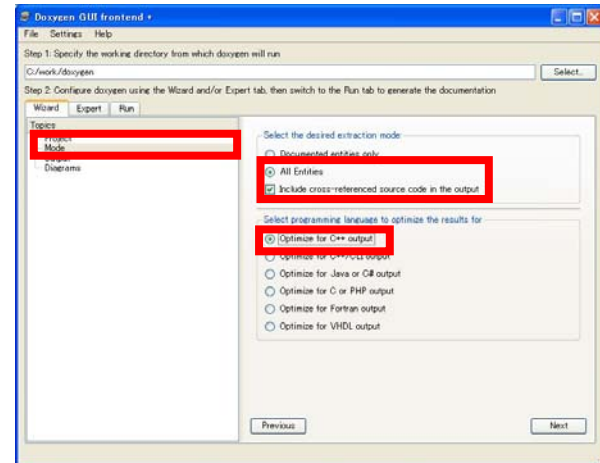


ドキュメント作成(Windows)

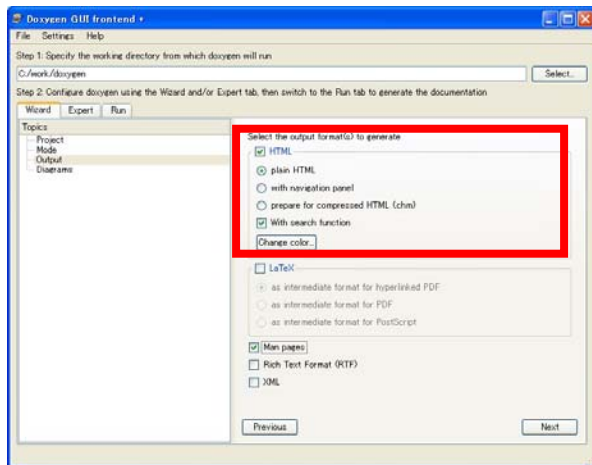
- ① Doxygen用GUIツールを起動
作業用ディレクトリ,ソース格納場所,
生成ファイル出力先,プロジェクト名を指定



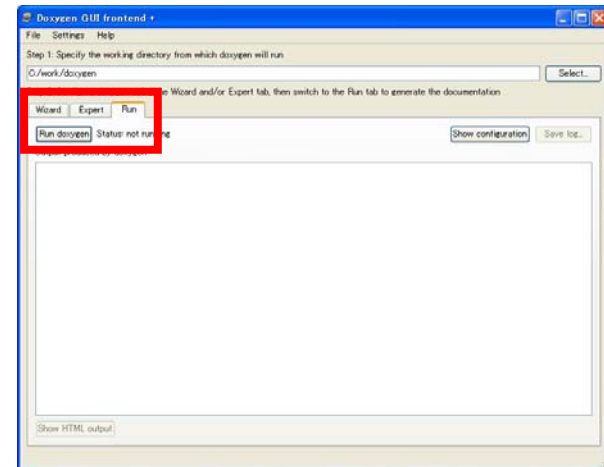
- ② 「Mode」セクションにて,
出力内容,使用言語を指定



- ③ 「Output」セクションにて, html出力を指定



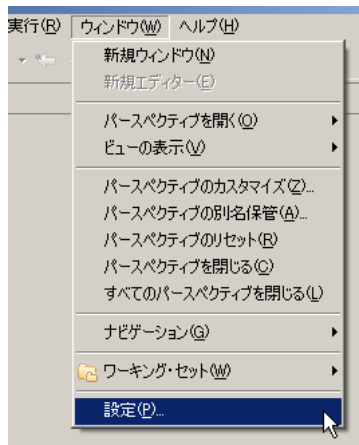
- ③ 「Run」タブにて, 「Run doxygen」を実行



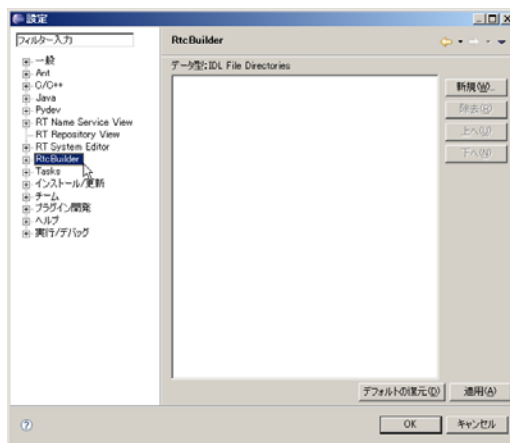
■ DataPortにて利用するデータ型の指定

→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

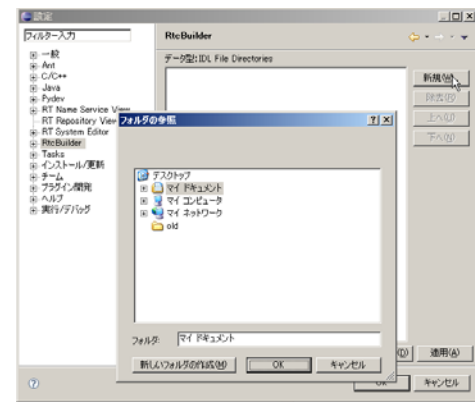
①メニューから「ウインドウ」→「設定」



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/