Programming with RTM

National Institute of Advanced Industrial Science and Technoloy

ICPS Research Center

Noriaki Ando

Verify Installation (Windows)

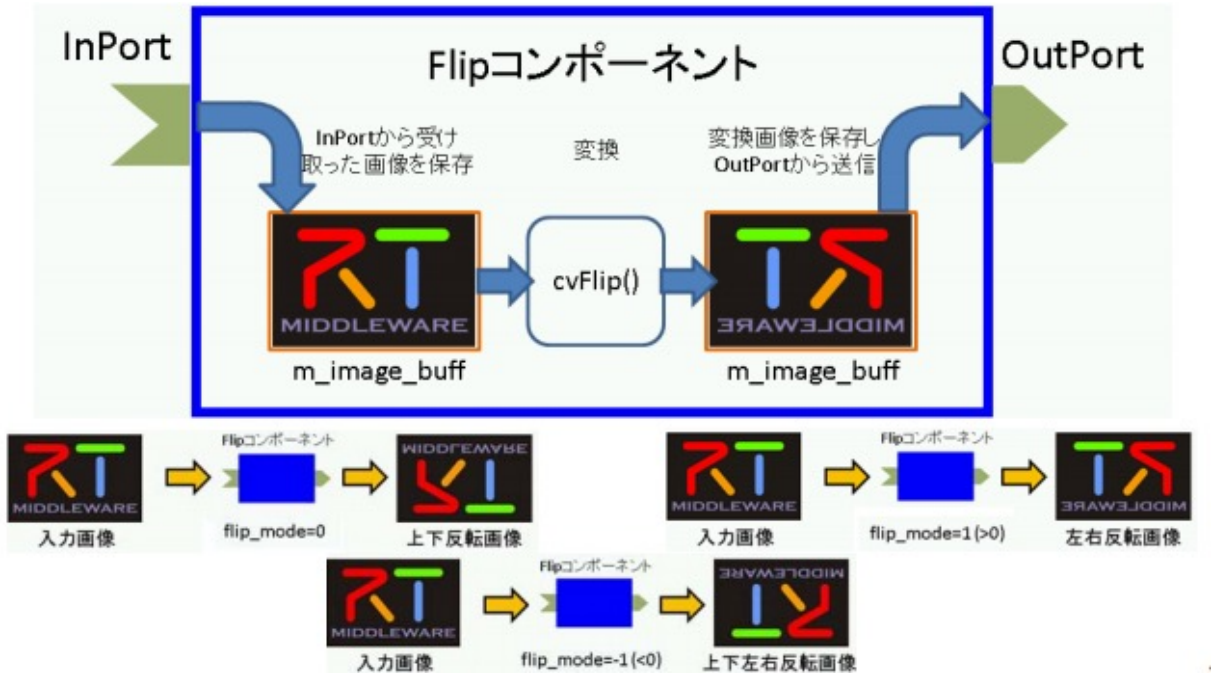
https://sealbreeder.github.io/TMU-Ubiquitous-Robotics/210519

- OpenRTM-aist
  - OpenRTM-aist-1.2.2-RELEASE_x86_64.msi (Use the 64-bit version)
  - Restart after installation
- Python
  - python-3.8
  - ※When installing the 32-bit version of OpenRTM-aist, Python also installs the 32-bit version. Python also installs the 64-bit version if you install openRTM-aist 64bit.
- CMake
  - cmake-3.20.x-win32-x86_64.msi
- Doxygen
  - doxygen-1.9.xx-setup.exe
- Visual Studio
  - Visual Studio 2019

If there is a newer version, install it

Practice contents

- Create a component to flip an image
  - Process image data received by InPort and output it from OutPort
  - Learn how to use data ports
  - Set the direction to be reversed by configuration parameters
    - Learn how to use configuration parameters
  - Rt System Editor connects to other RTCs and activates RTC
    - Learn how to use the RT System Editor



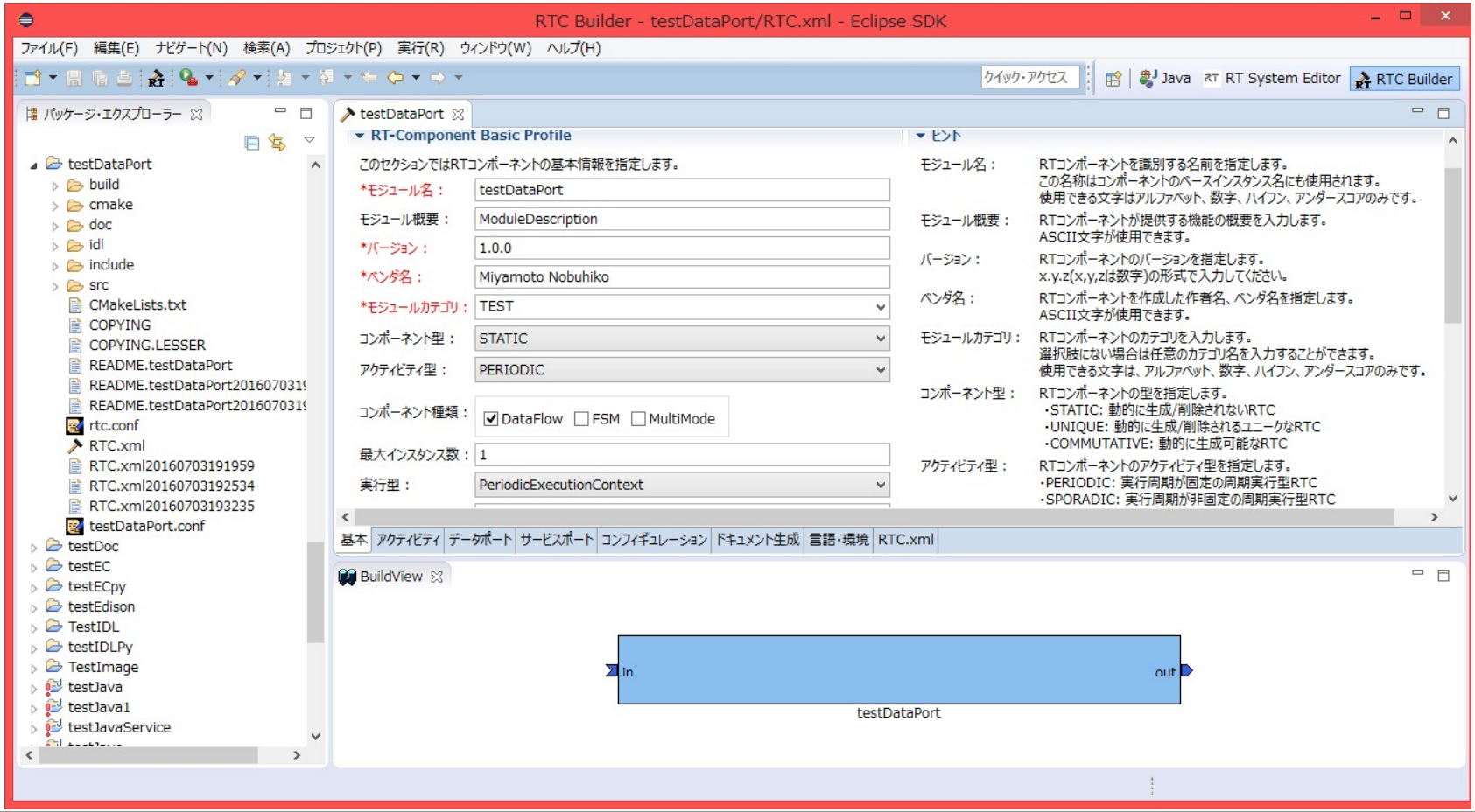
The whole procedure

- Creation of model types such as source code by RTC Builder
  - Edit and build source code
  - Generate various files required for build
    - Edit .txt CMakeLists
    - Various files generation by Cmake
  - Edit source code
    - Edit Flip.h
    - Edit .cpp Flip
  - Build
    - Visual Studio、Code::Blocks
  - Rt system creation and operation confirmation by RT system editor
    - RT system creation
    - Data port connection, configuration parameter settings

Component development tools RTCBuilder

RTCBuilder

- A tool for inputting component profile information and generating model types such as source code
  - Output C++, Python, and Java source code
  -

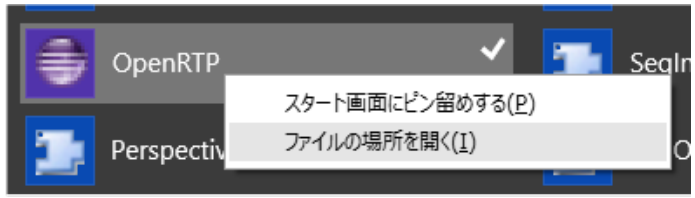


Launch RTC Builder

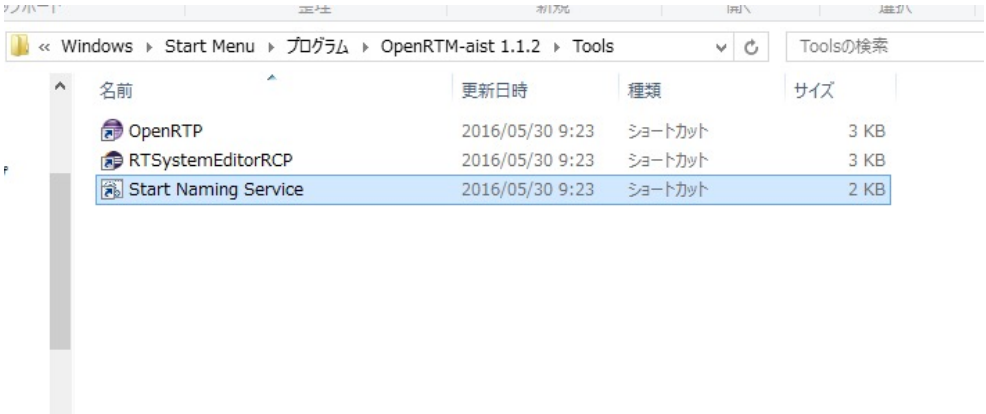
- Steps to start
  - Windows 10
    - Enter "OpenRTP" in the lower left search window
  - Ubuntu
    - Go to the directory where Eclipse was deployed and click the following command
    - $ ./openrtp

Launch RTC Builder

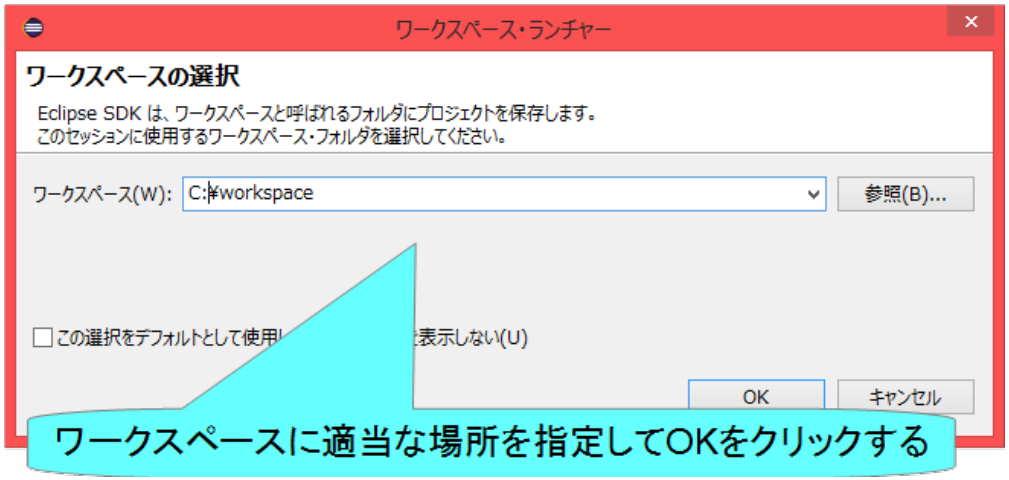
- It can be very time consuming to start from the app view, so it is recommended that you open the Start menu folder by doing the following:
-



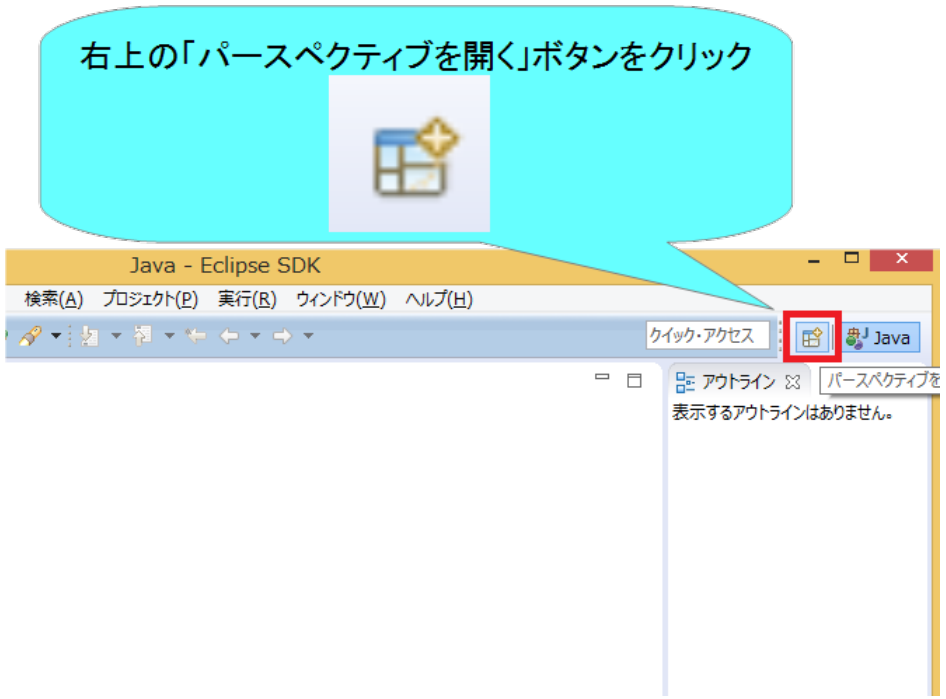
Start naming Serviceを右クリックして
「ファイルの場所を開く」を選択

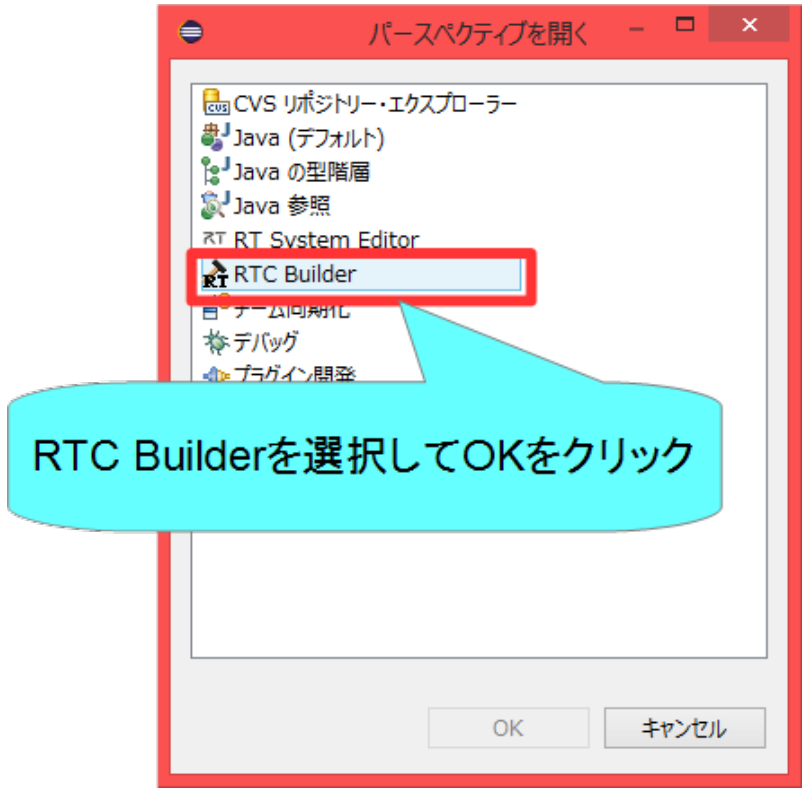


Launch RTC Builder



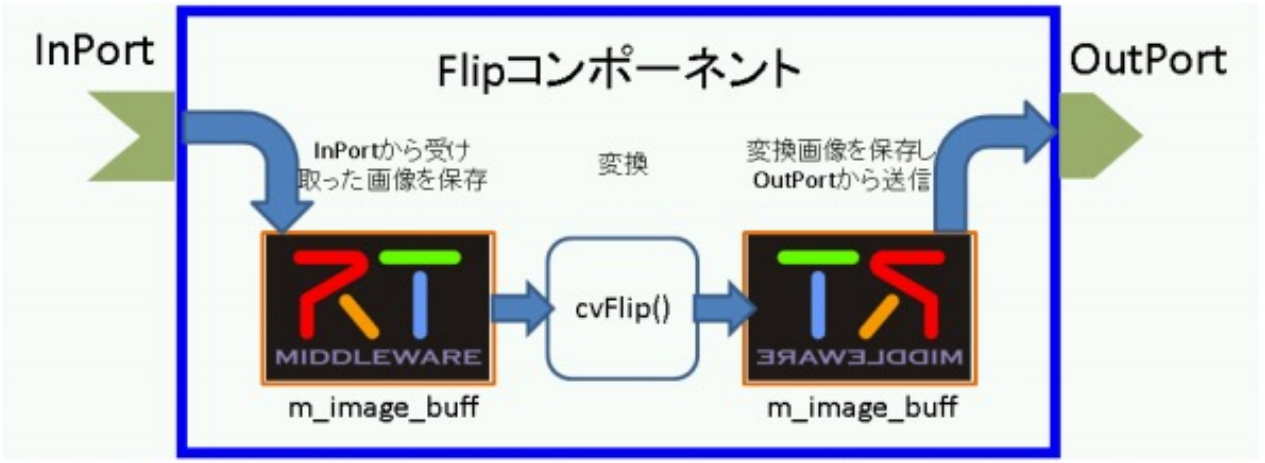
Launch RTC Builder





Project creation

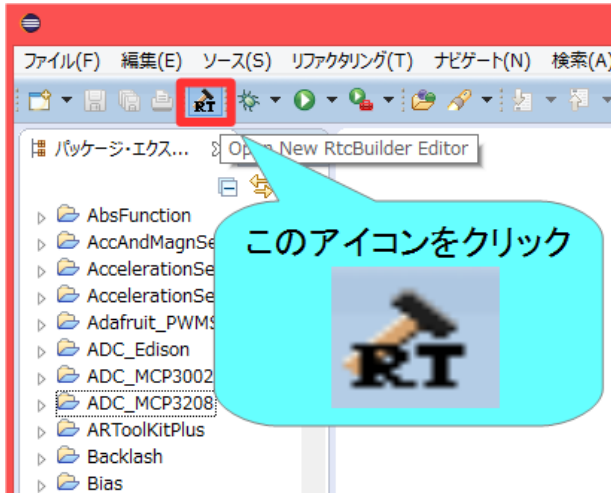
- Create skeleton code for Flip components.
  - Components that invert images
    - Process image data received by InPort and output it from OutPort
    - Set the direction to be reversed by configuration parameters

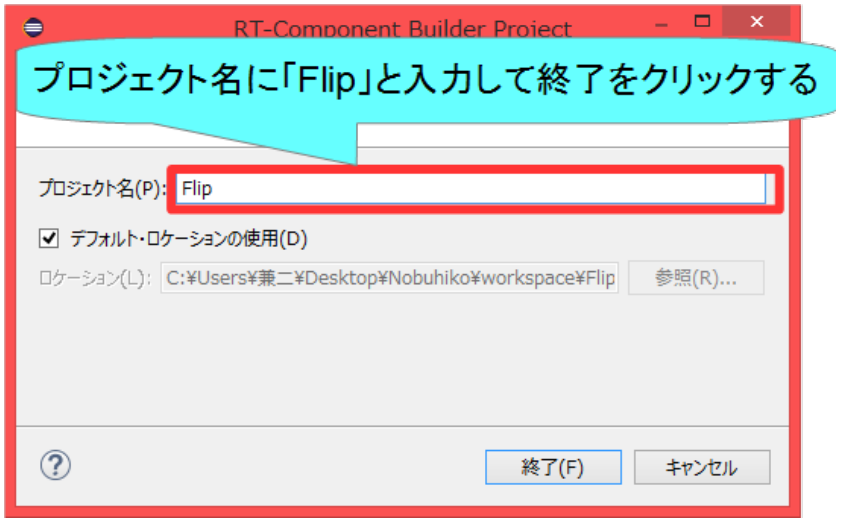


Tutorial Page

- Open a page from the Official OpenRTM-aist website as shown on the right
  - https://bit.ly/2LU3JNZ

Project creation

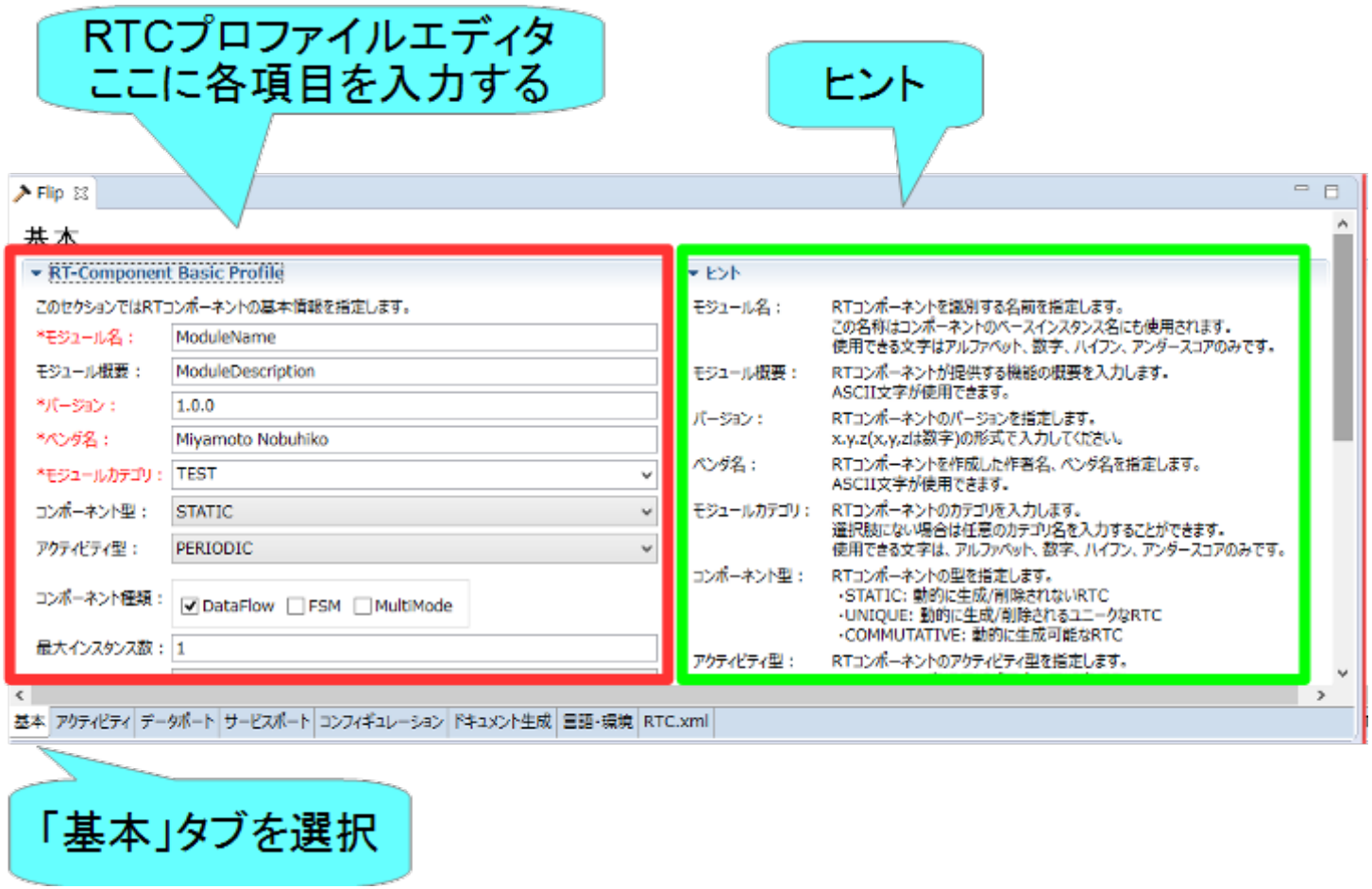




- A folder called "Flip" is created in the directory specified in the workspace when Eclipse starts
  - At this point, only .xml RTC" and ".project" are generated
- Set up the following items
  - Base profile
  - Activity profile
  - Data port profiles
  - Service port profile
  - configuration
  - document
  - Language environment
  - RTC.xml
  -

Enter a base profile

- Set basic information for components, such as rt component profile information.
- Code generation, import/export, and packaging
-

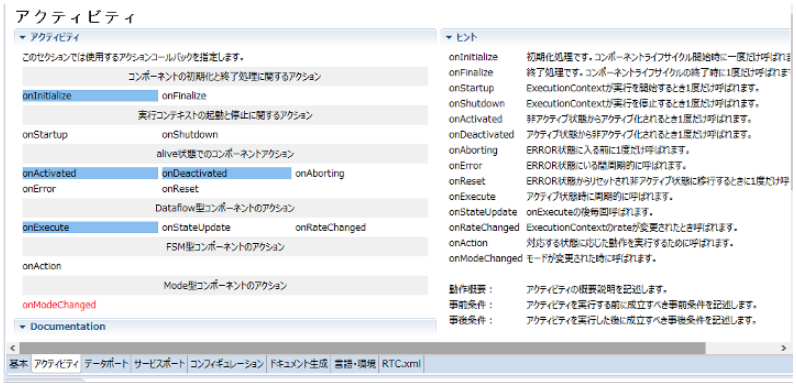


Enter a base profile

- Module name
  - Flip
- Module Overview
  - Optional (Flip image component)
- Version
  - Optional (1.0.0)
- Vendor name
  - Arbitrary
- Module category
  - Optional (ImageProcessing)
- Component type
  - STATIC
- Activity type
  - PERIODIC
- Component type
  - DataFlow
- Maximum number of instances
  - 1
- Execution type
  - PeriodicExecutionContext
- Execution cycle
  - 1000.0
- Summary
  - arbitrary

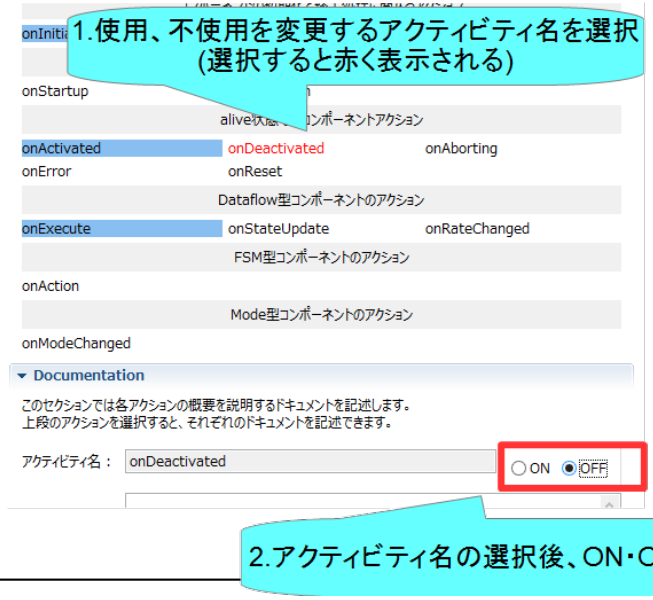
アクティビティの設定

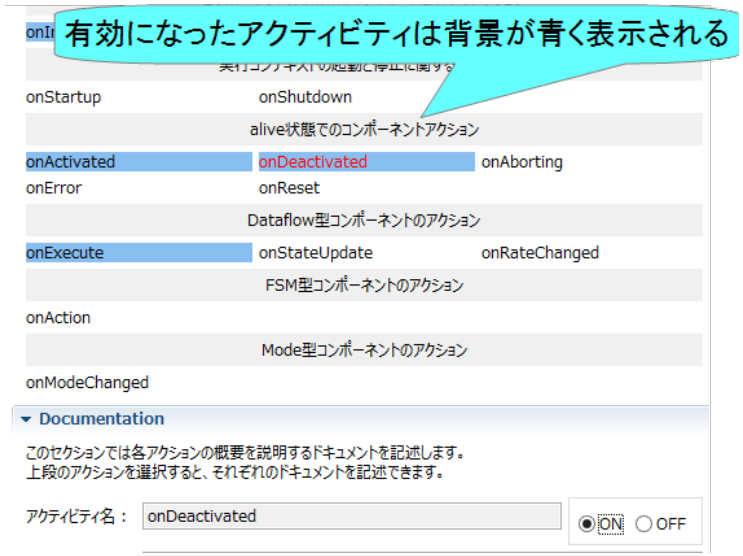
- 使用するアクティビティを設定する



「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



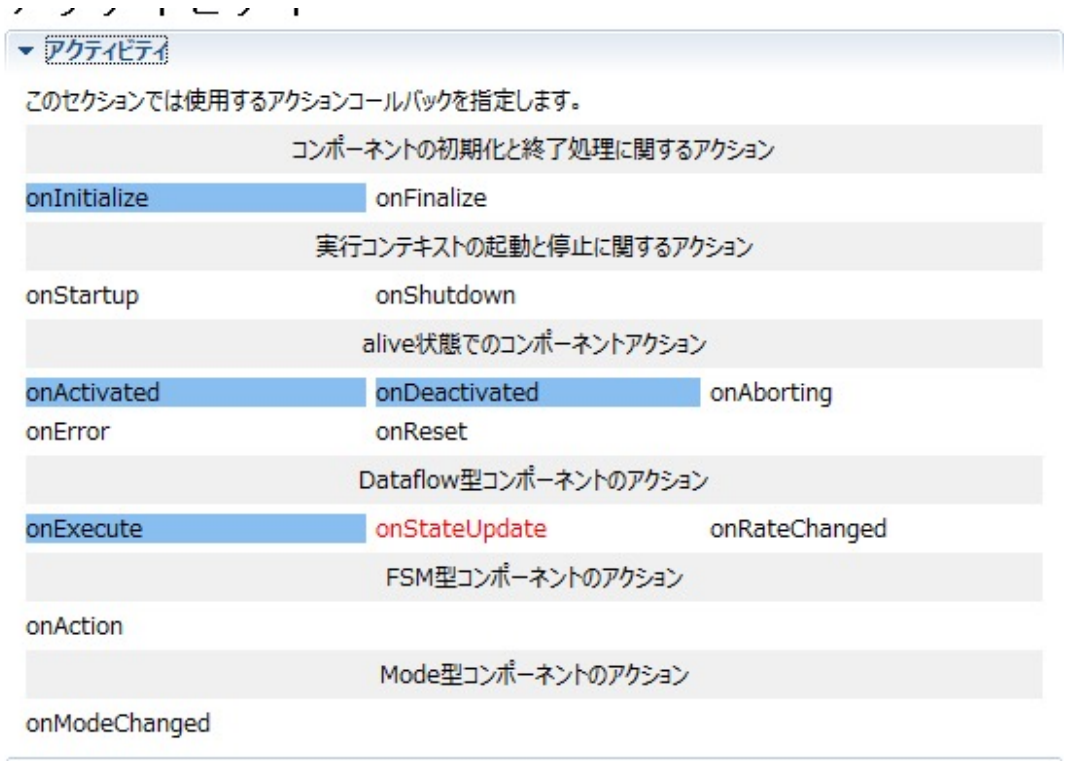


Activity settings

Callback Functions	Process
onInitialize	Initialization process
onActivated	Called only once when activated
onExecute	Called periodically when active
onDeactivated	Called only once when deactivated
onAborting	Called only once before entering ERROR state
onReset	Called only once when reset
onError	Called periodically in ERROR state
onFinalize	Called only once at the end
onStateUpdate	Called every time after onExecute
onRateChanged	Called only once when the ExecutionContext rate changes
onStartup	Called only once when ExecutionContext starts execution
onShutdown	Called only once when ExecutionContext stops executing

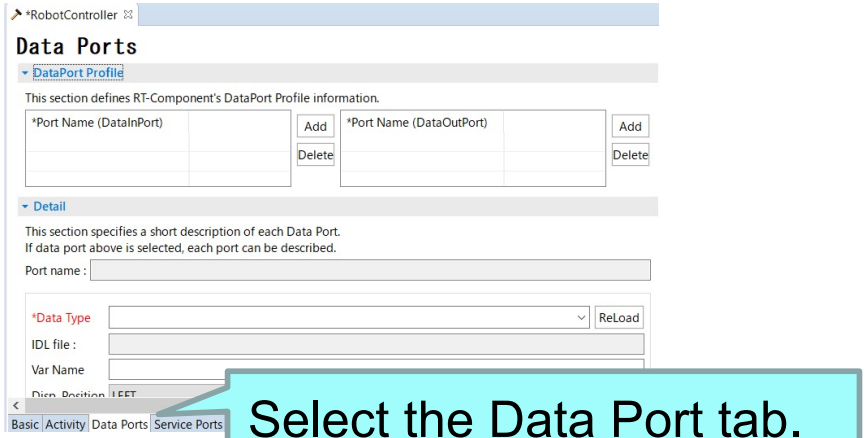
Activity settings

- Enable the following activities
  - onInitialize
  - **onActivated**
  - **onDeactivated**
  - **onExecute**

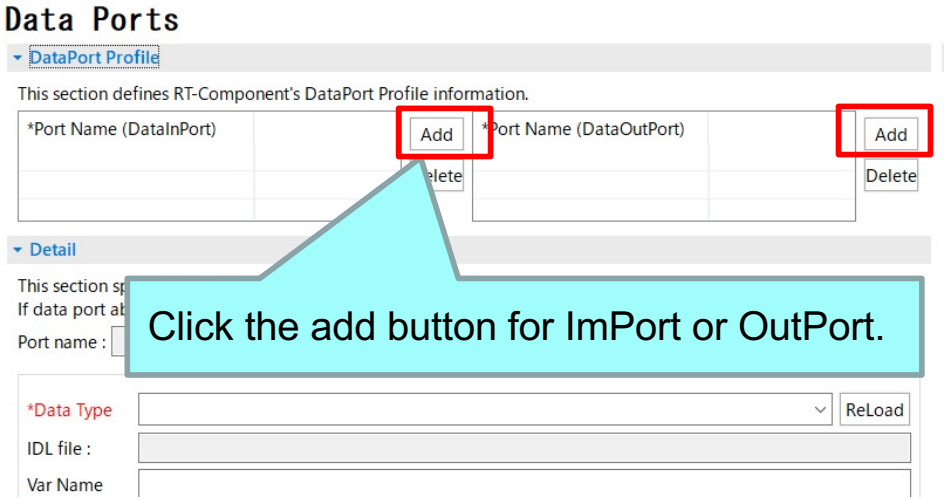


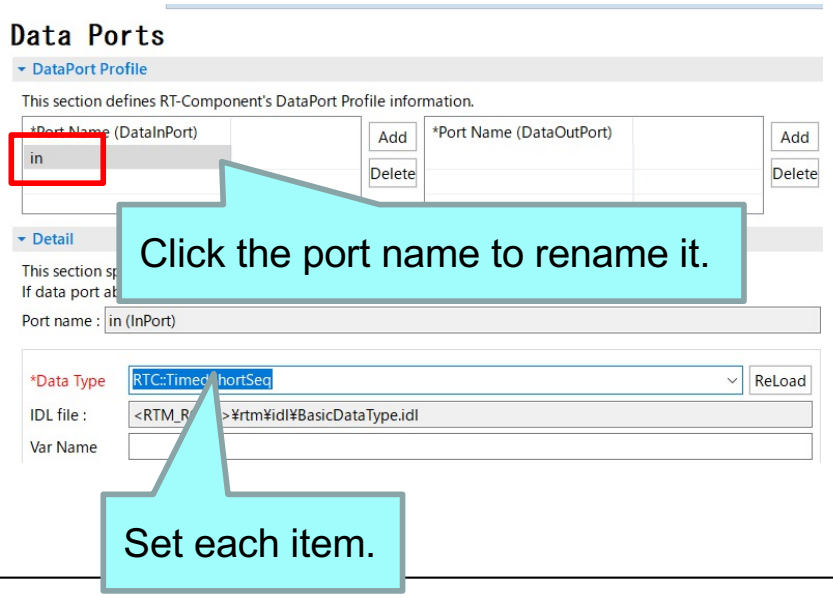
Data port settings

- Add and set InPort and OutPort



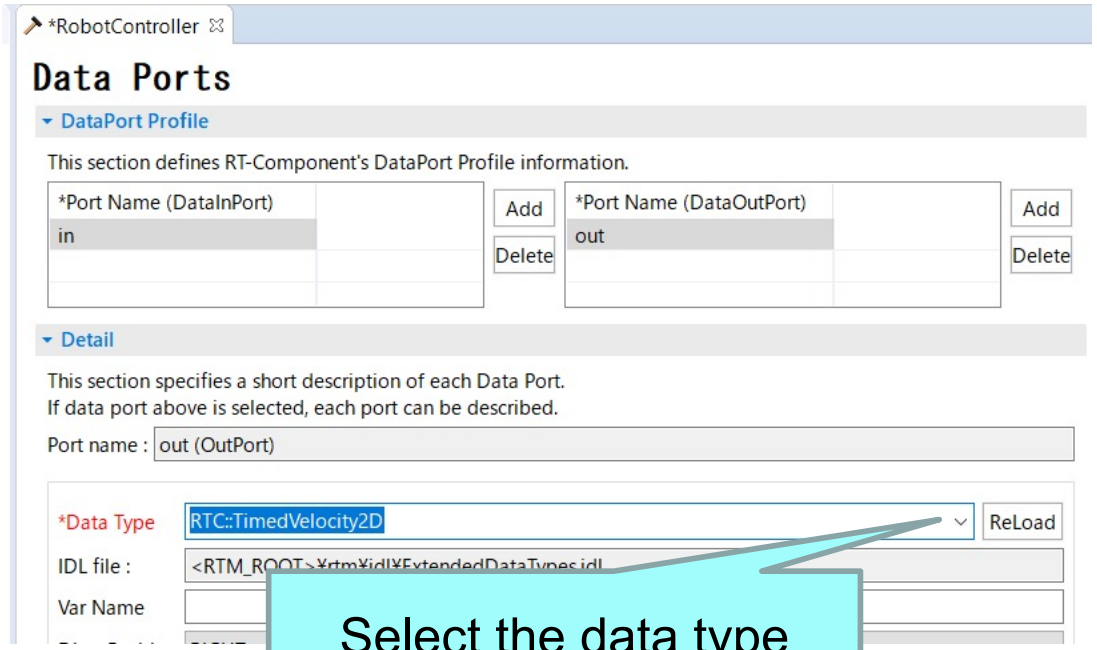
- Steps to add a data port





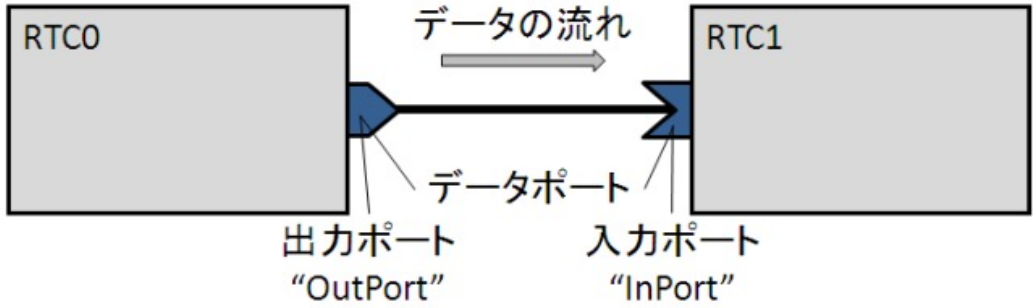
Data port settings

- Set the following InPort
  - **in**
    - Data Type: **RTC::TimedShortSeq**
    - Please do not mistake it for TimedShort type.
- Set the following OutPort
  - **out**
    - Data Type : **RTC::TimedVelocity2D**
    - Please do not mistake it for TimedVelocity3D type and TimedVector2D.

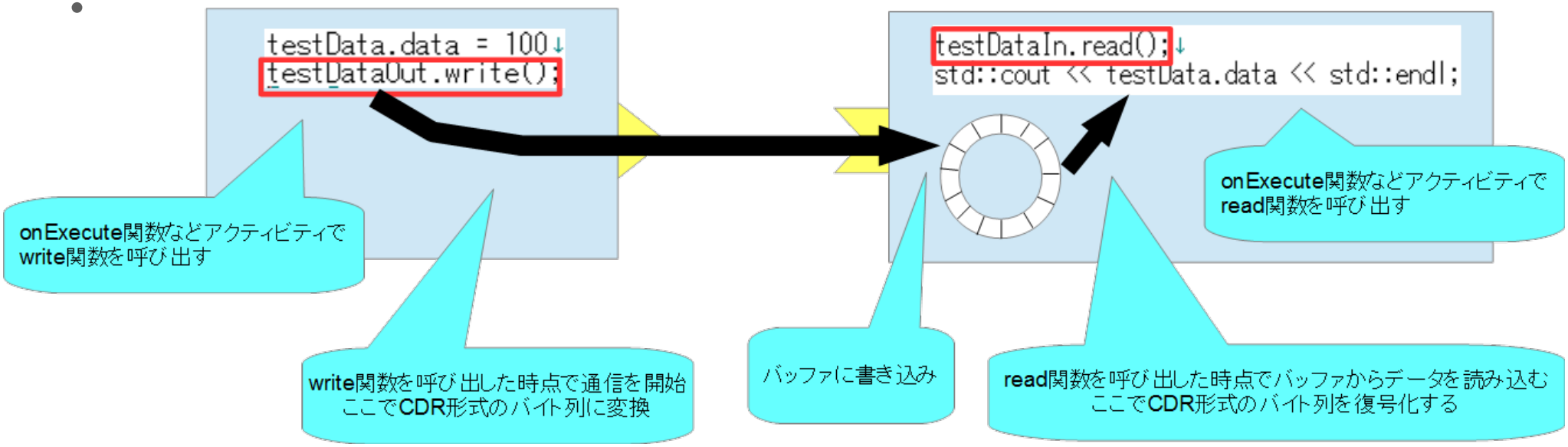


About the data port

- Port for communicating continuous data

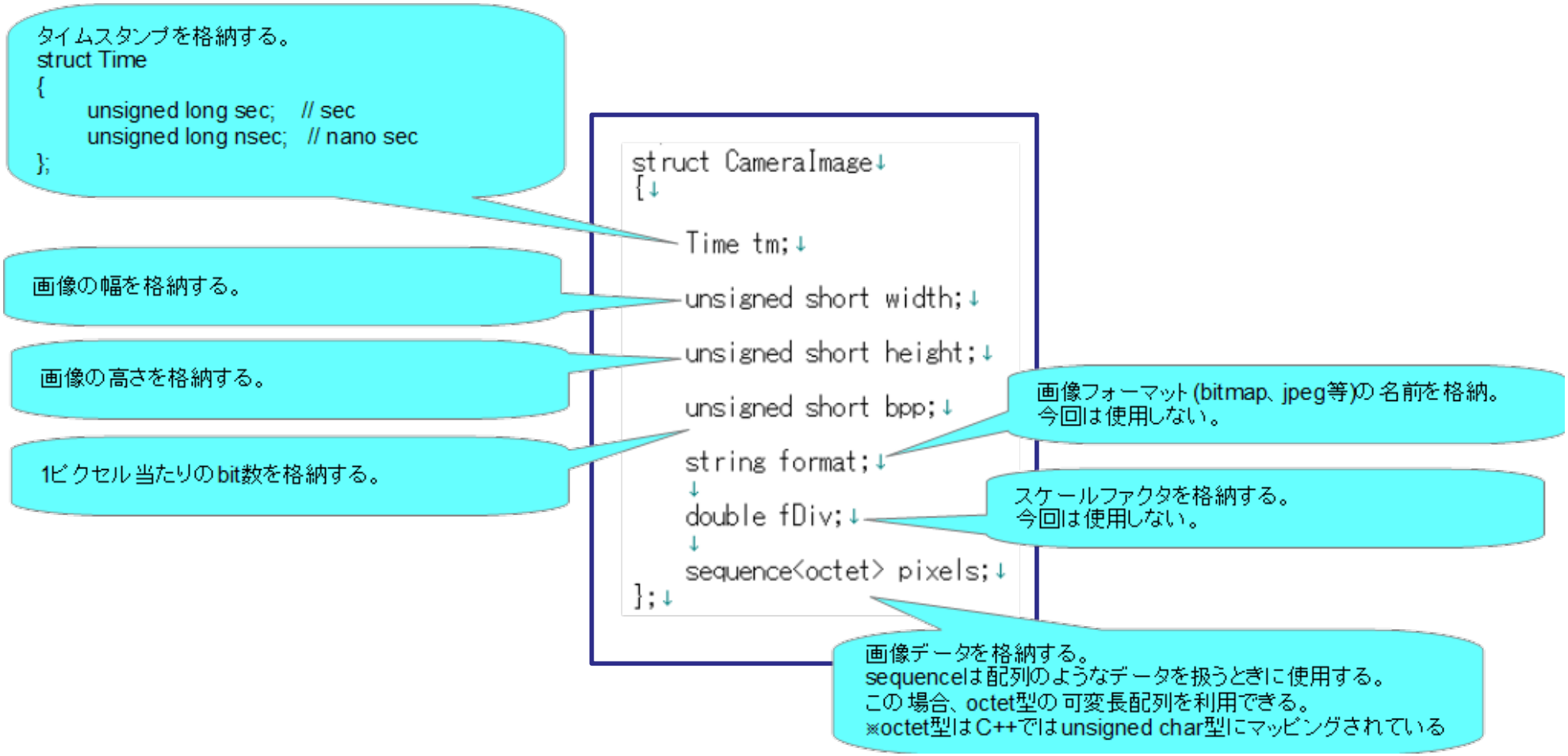


- The following examples are dataflow type push, subscription type is flush, interface type corba_cdr type is
-



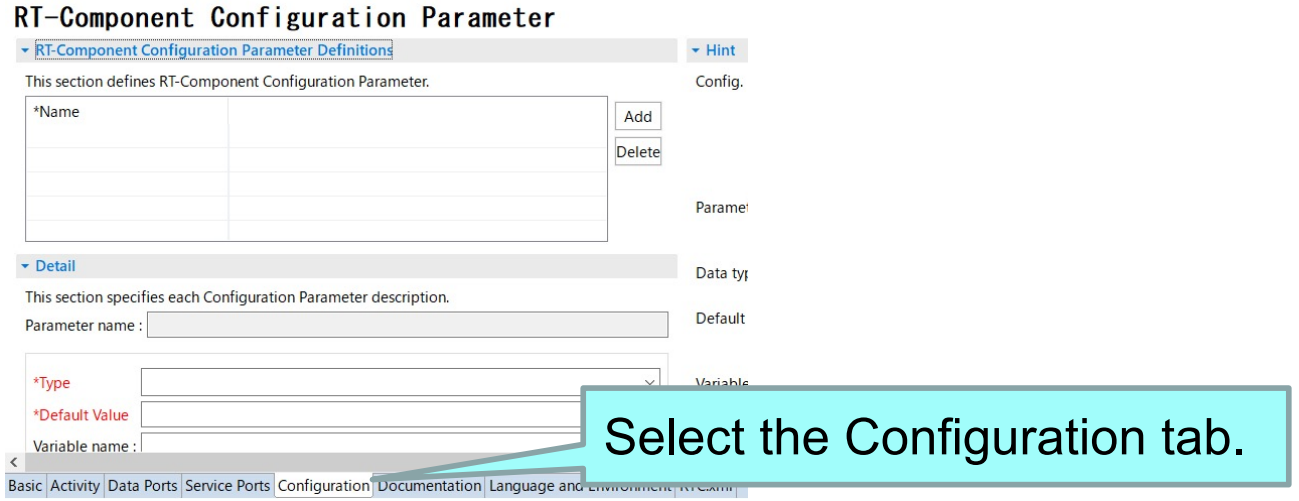
About RTC::CameraImage type

- Data types for image data communication defined in InterfaceDataTypes.idl
-

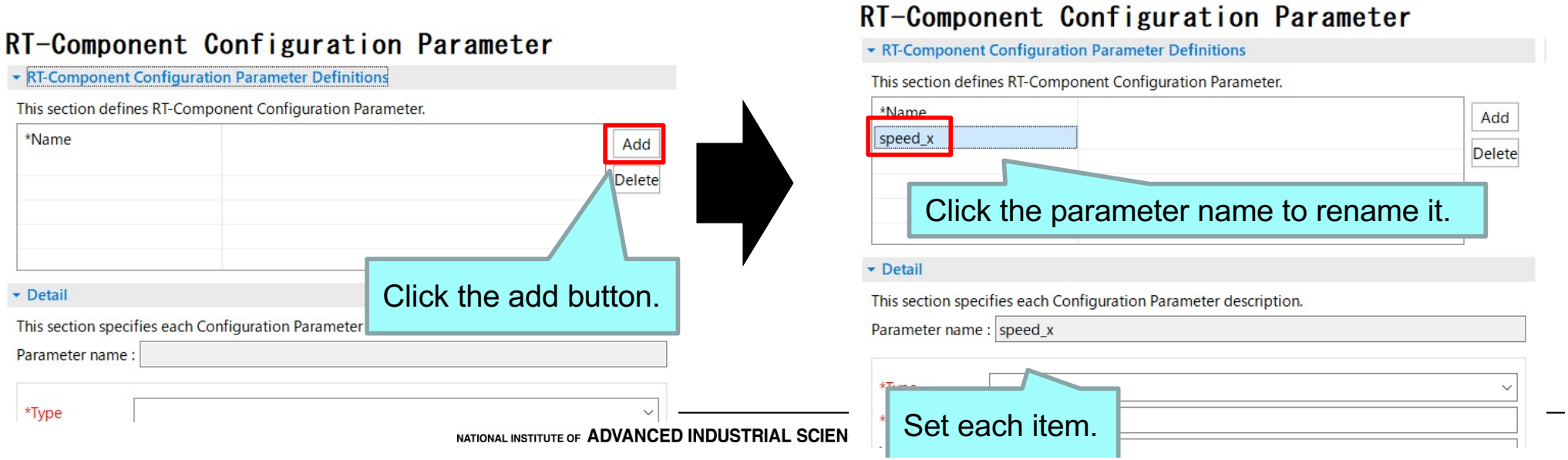


Configuration Parameter settings

- Add and set configuration parameters

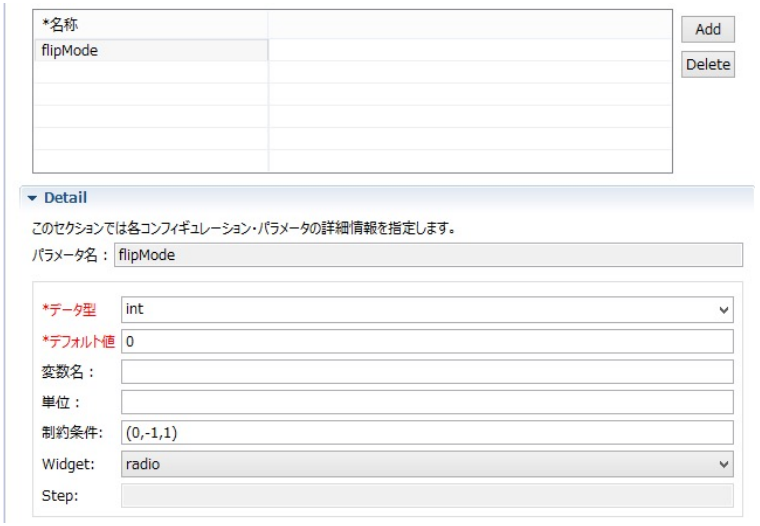


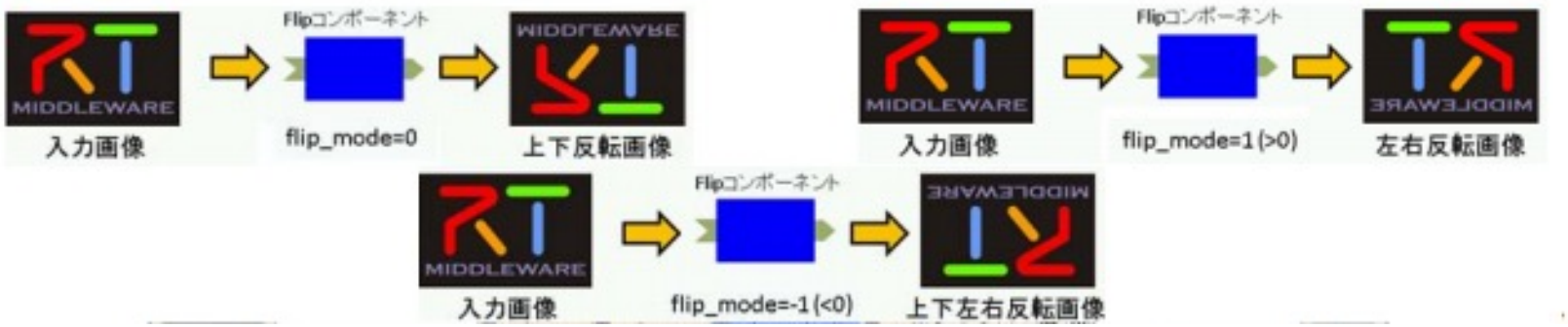
- Steps to add configuration parameters



Configuration Parameter settings

- Set up the following configuration parameters
- flipMode
  - Data type : int
  - Default value : 0
  - Constraint : (0,-1,1)
  - Widget : radio
  - Other items are optional
  -





- Make the direction to flip configurable
-

Configuration parameter constraints, and widget settings

- Show GUI when editing configuration parameters in RT System Editor

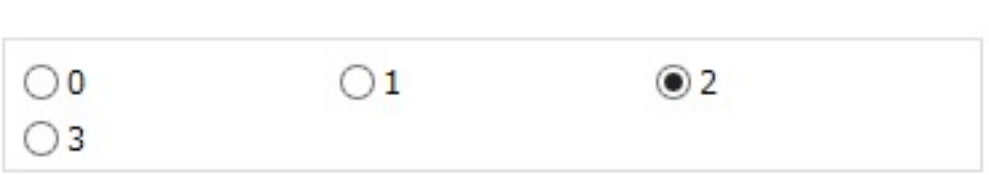
- Widget : text

- Constraint : 0<=x<=100
- Widget : spin
- Step : 10

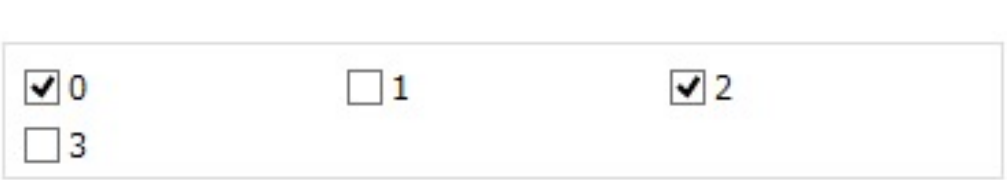
- Constraint : 0<=x<=100
- Widget : slider
- Step : 10

Configuration parameter constraints, and widget settings

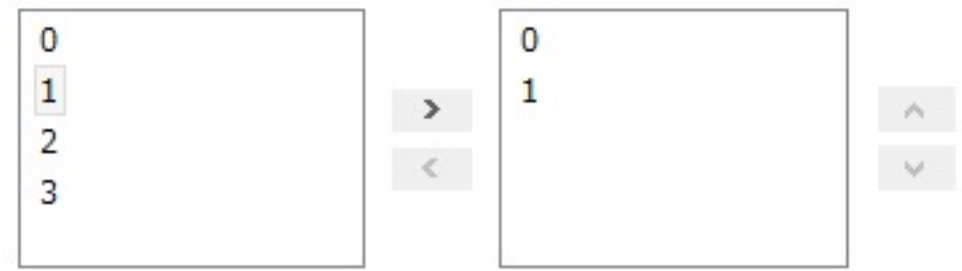
- Constraint : (0,1,2,3)
- Widget : radio



- Constraint : (0,1,2,3)
- Widget : checkbox

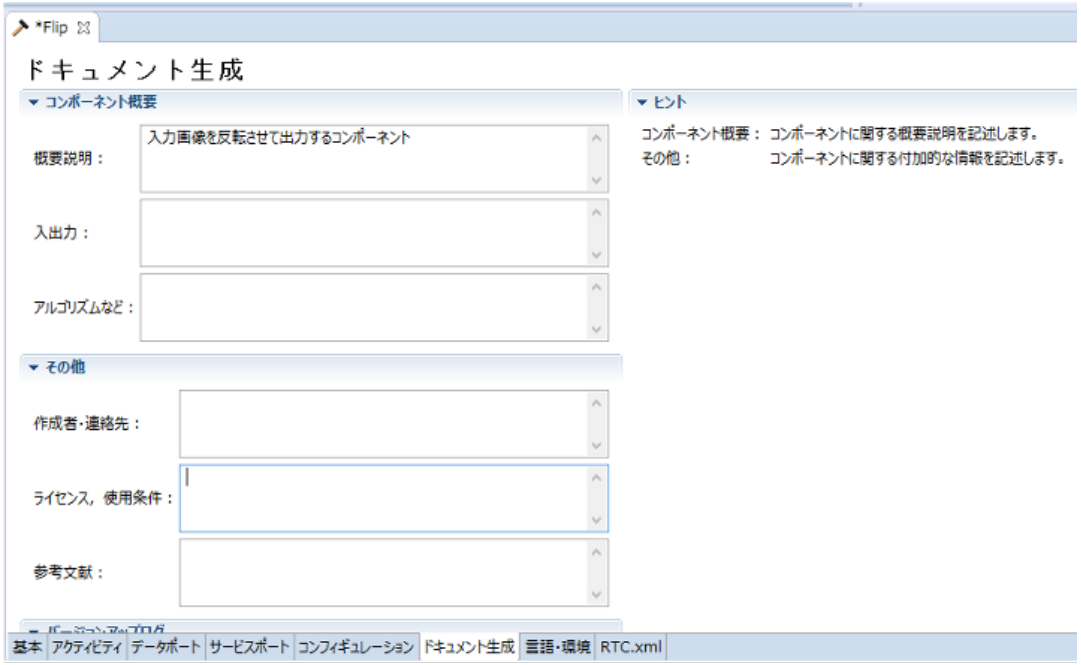


- Constraint : (0,1,2,3)
- Widget : ordered_list



Document settings

- Set up various document information
-

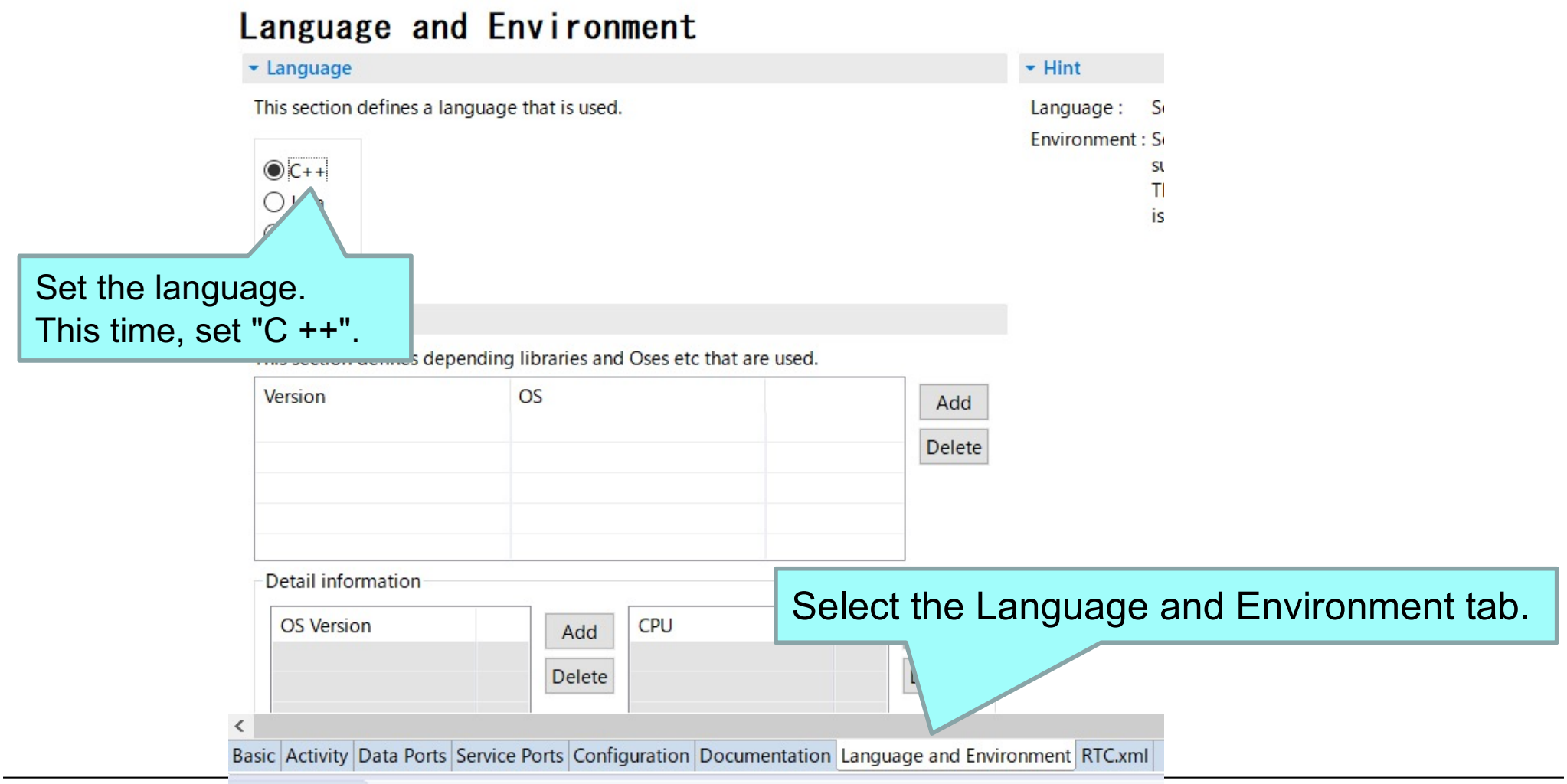


「ドキュメント生成」タブを選択

- Please set it appropriately this time.
  - You can leave it blank

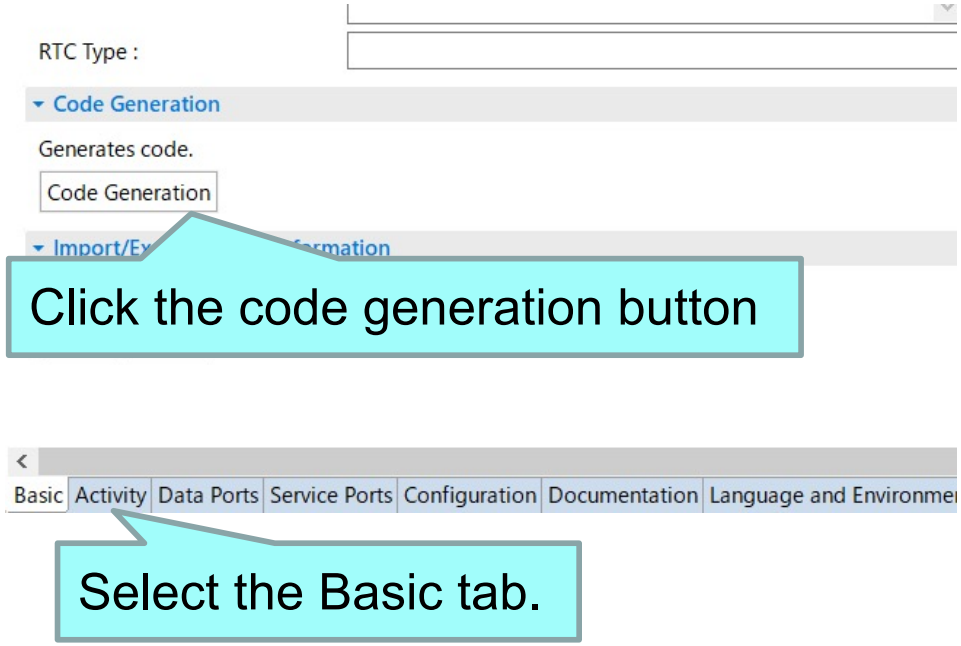
Language setting

- Set information about the language to be implemented and the operating environment.

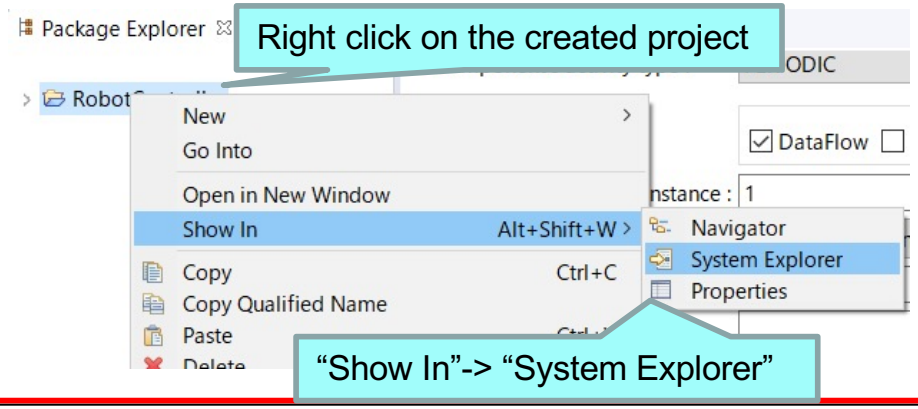


Skeleton code generation

- A skeleton code is generated by pressing the code generation button from the basic tab.
  - Workspace¥RobotController
    - Source code
      - C++ Source files(.cpp)
      - Header files (.h)
    - CMakeLists.txt
    - rtc.conf、RobotController.conf
    - etc.



- Check the generated file
  - Right-click the created project and select “Show In“(表示方法)-> "System Explorer".
  - Explorer will open the workspace folder, so check if the above file exists



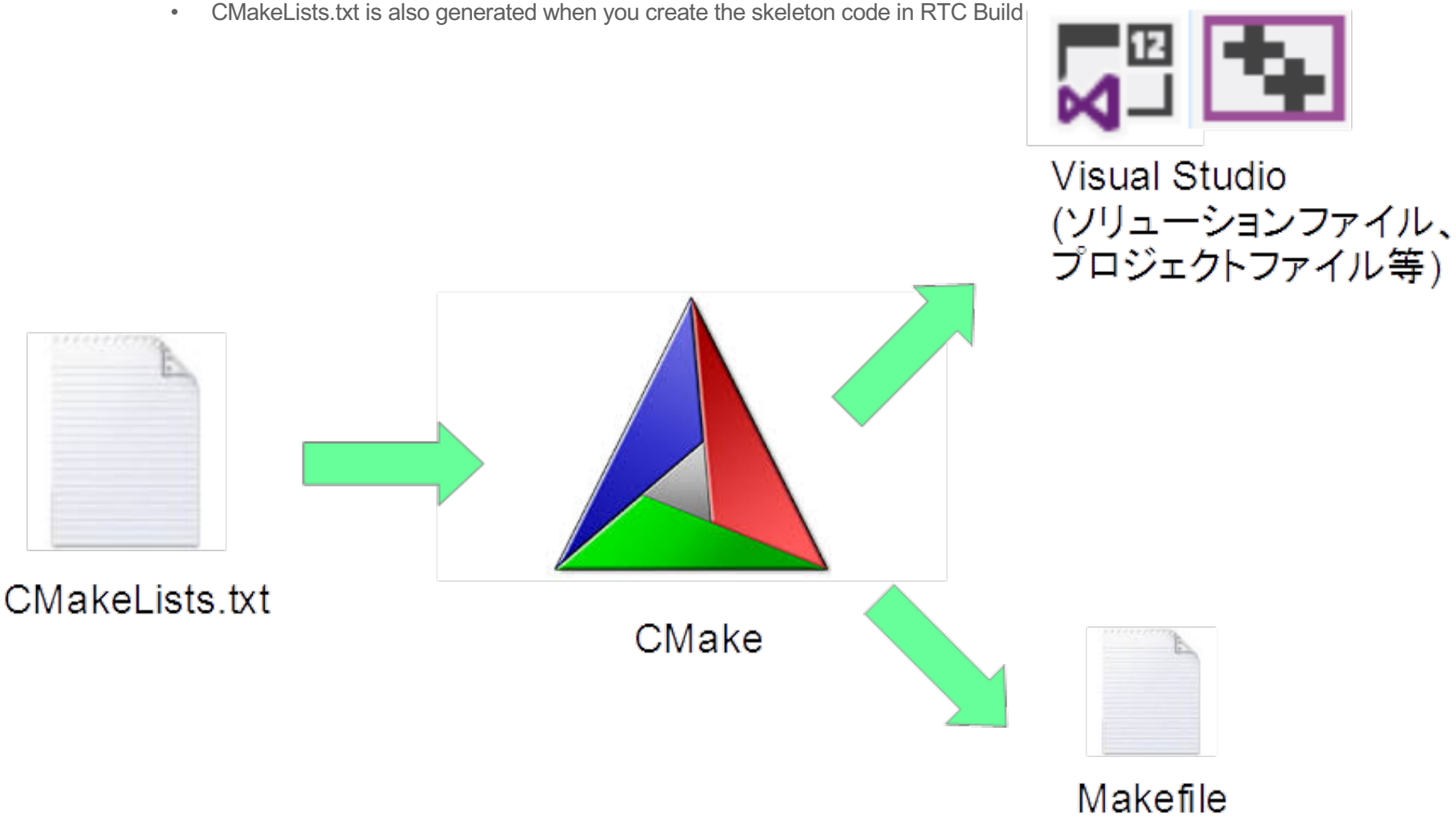
Edit source code, build RTC

Steps to build

- Generate various files required for build
  - Various files generated by CMake
- Edit the source code
  - Edit RobotController.h
  - Edit RobotController.cpp
- Build
  - Windows：Visual Studio
  - Ubuntu：Code::Blocks

CMake

- Generate various files required for build
  - Describe the settings in CMakeLists.txt.
    - CMakeLists.txt is also generated when you create the skeleton code in RTC Build



Editing CMakeLists.txt

- Modify CMakeLists.txt to use OpenCV
  - Open **CMakeLists.txt in the** worksapce¥Flip directory by Eclipse or notepad and modify as follows

```
1  set(comp_srcs Flip.cpp )
2  set(standalone_srcs FlipComp.cpp)
3
4  find_package(OpenCV REQUIRED)
5
6  if (DEFINED OPENRTM_INCLUDE_DIRS)
7    string(REGEX REPLACE "-I" "" ";"
8      OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")
```

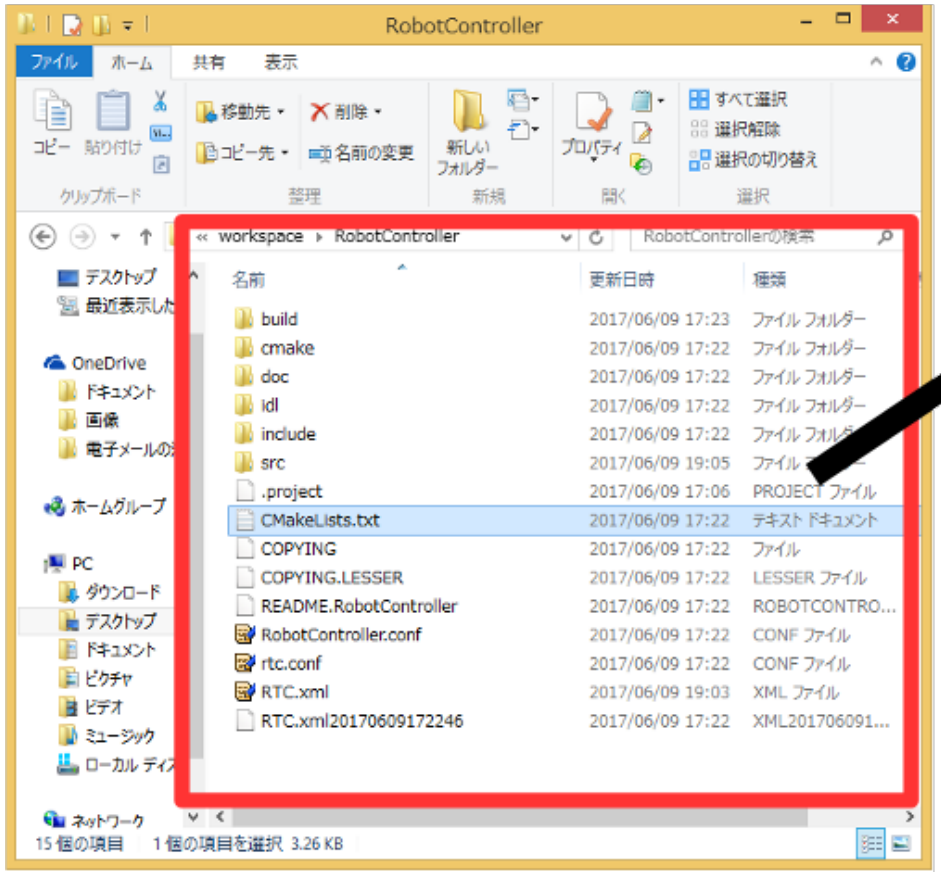
find_package(OpenCV REQUIRED)
を追加

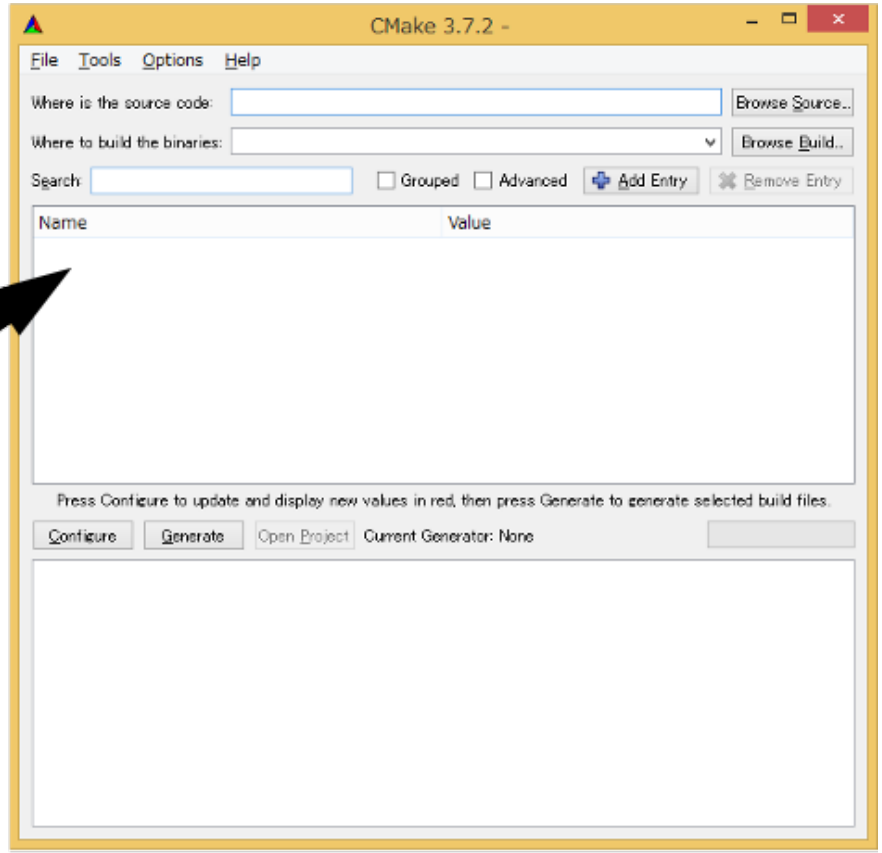
```
47 endif(NOT TARGET ALL_IDL_TGT)
48 add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
49 target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES} ${OpenCV_LIBS})
50
51 add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
52   ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
53 target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES} ${OpenCV_LIBS})
54
```

target_link_librariesの引数に${OpenCV_LIBS}を追加
※2箇所あるので注意

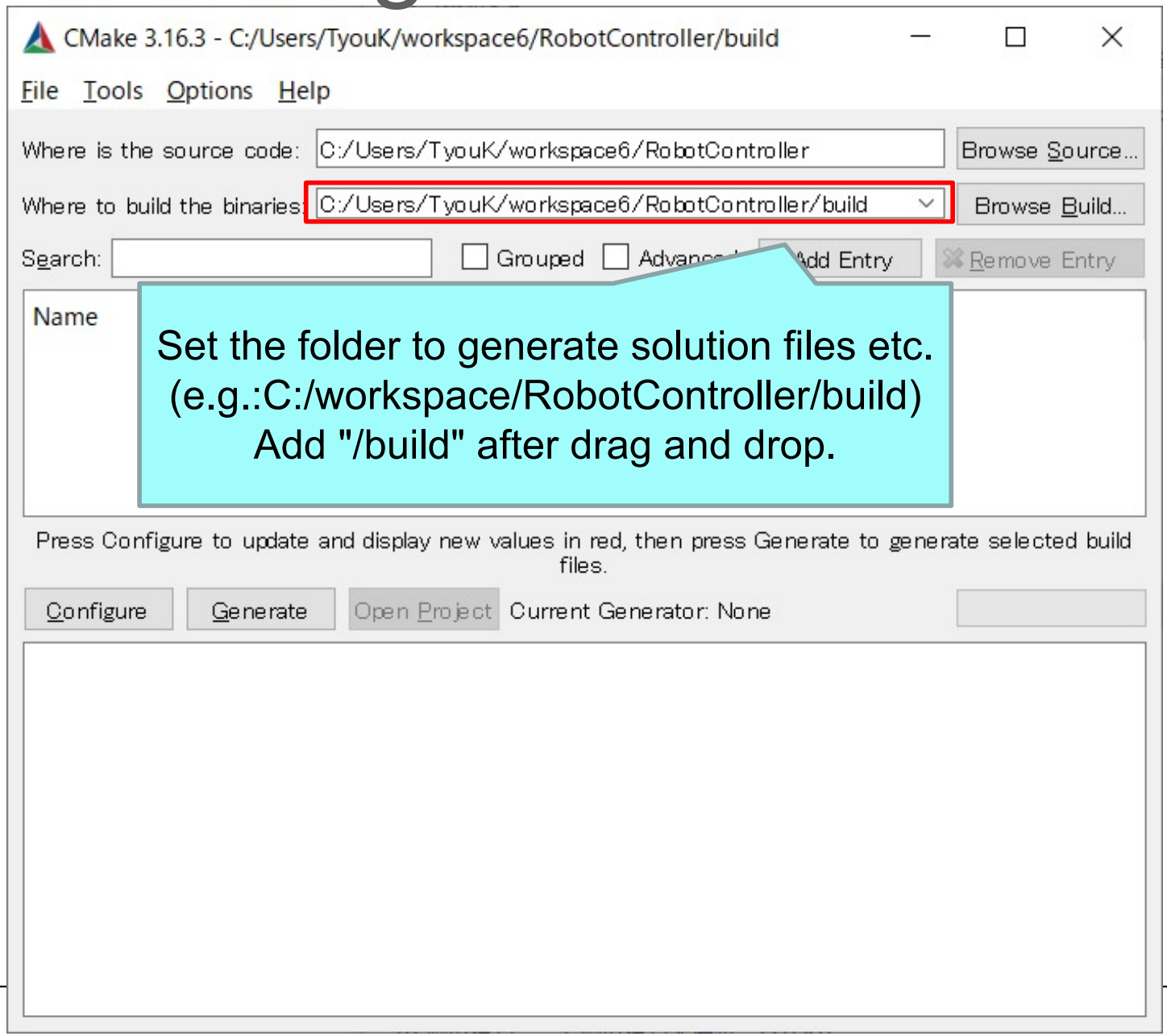
Generating files needed for build

- Drag and drop CMakeLists.txt onto cmake-gui
  - CMakeLists.txt is a folder for projects generated by RTC Builder.
    (例：C:¥workspace¥RobotController)

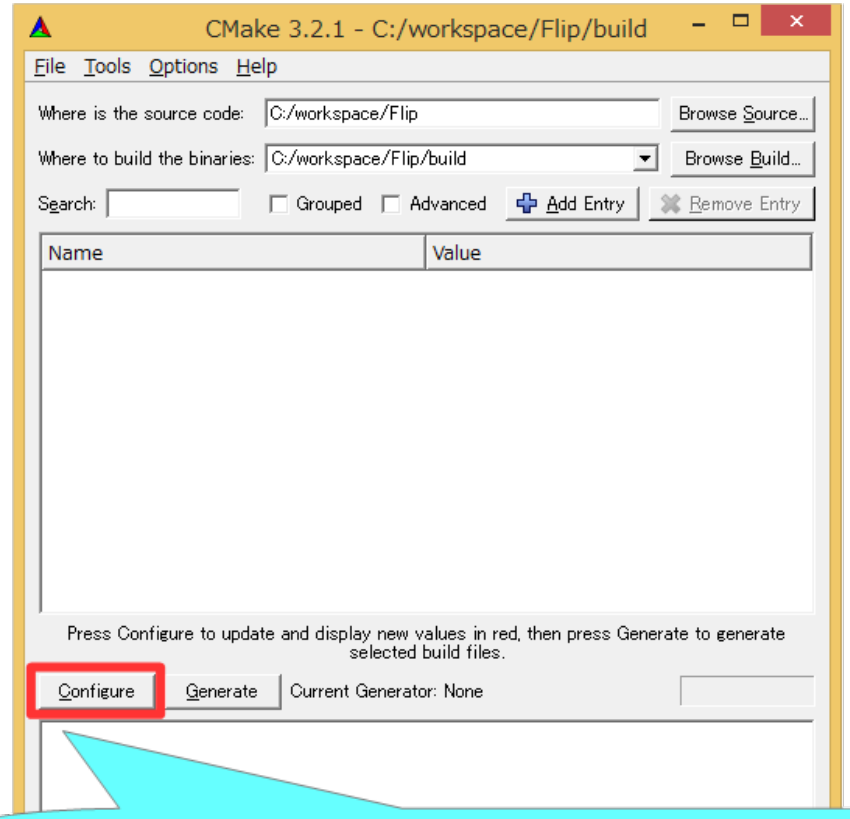




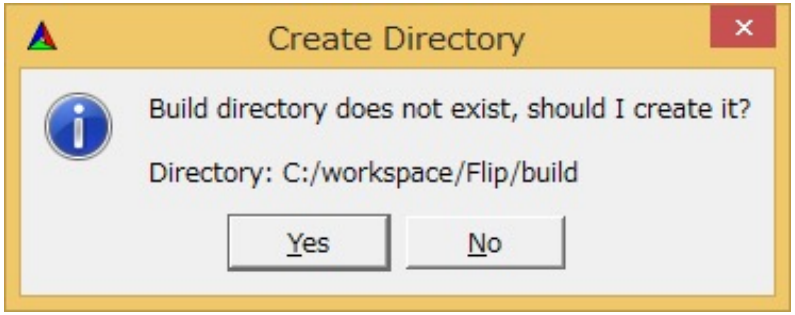
Generating files needed for build



Generate files needed for build



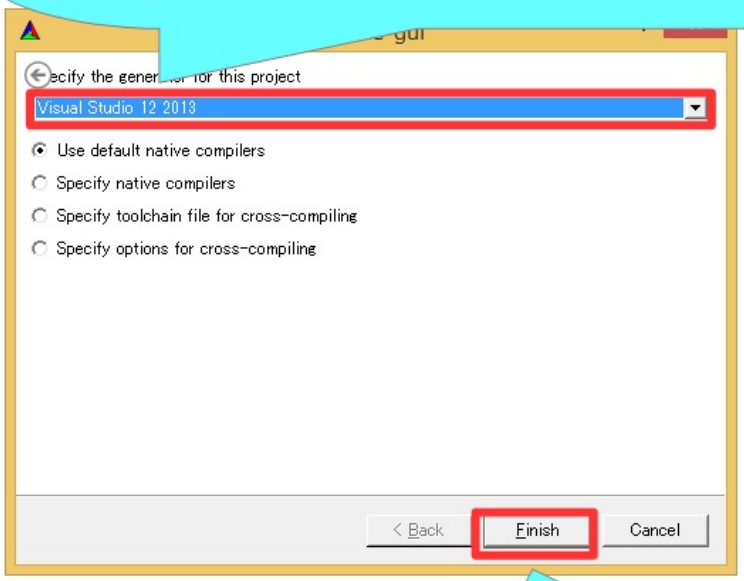
Configureボタンを押す
コンパイルに必要な情報の収集(必要なライブラリの検出など)を行う



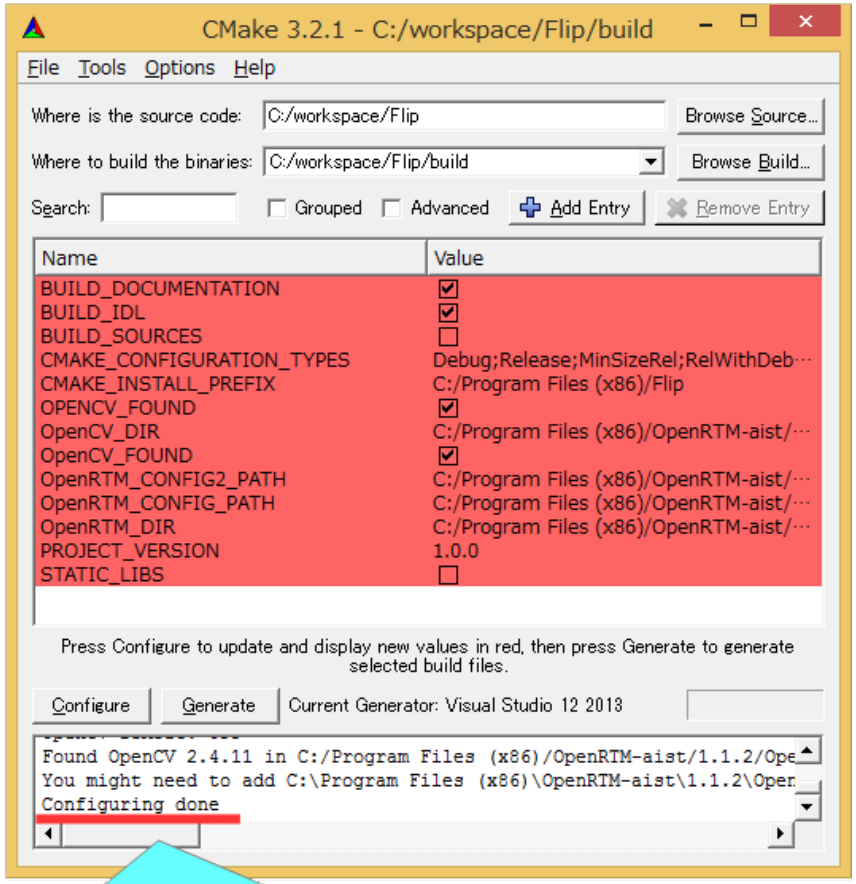
buildフォルダが存在しない場合は
作成するかどうかきかれるため「Yes」を選択

Generate files needed for build

ビルド環境の設定
Visual Studio 2010 32bit → Visual Studio 10 2010
Visual Studio 2012 32bit → Visual Studio 11 2012
Visual Studio 2012 64bit → Visual Studio 11 2012 Win 64
Visual Studio 2013 32bit → Visual Studio 12 2013
Visual Studio 2013 64bit → Visual Studio 12 2013 Win 64
Code::Blocks → CodeBlocks -Unix Makefiles

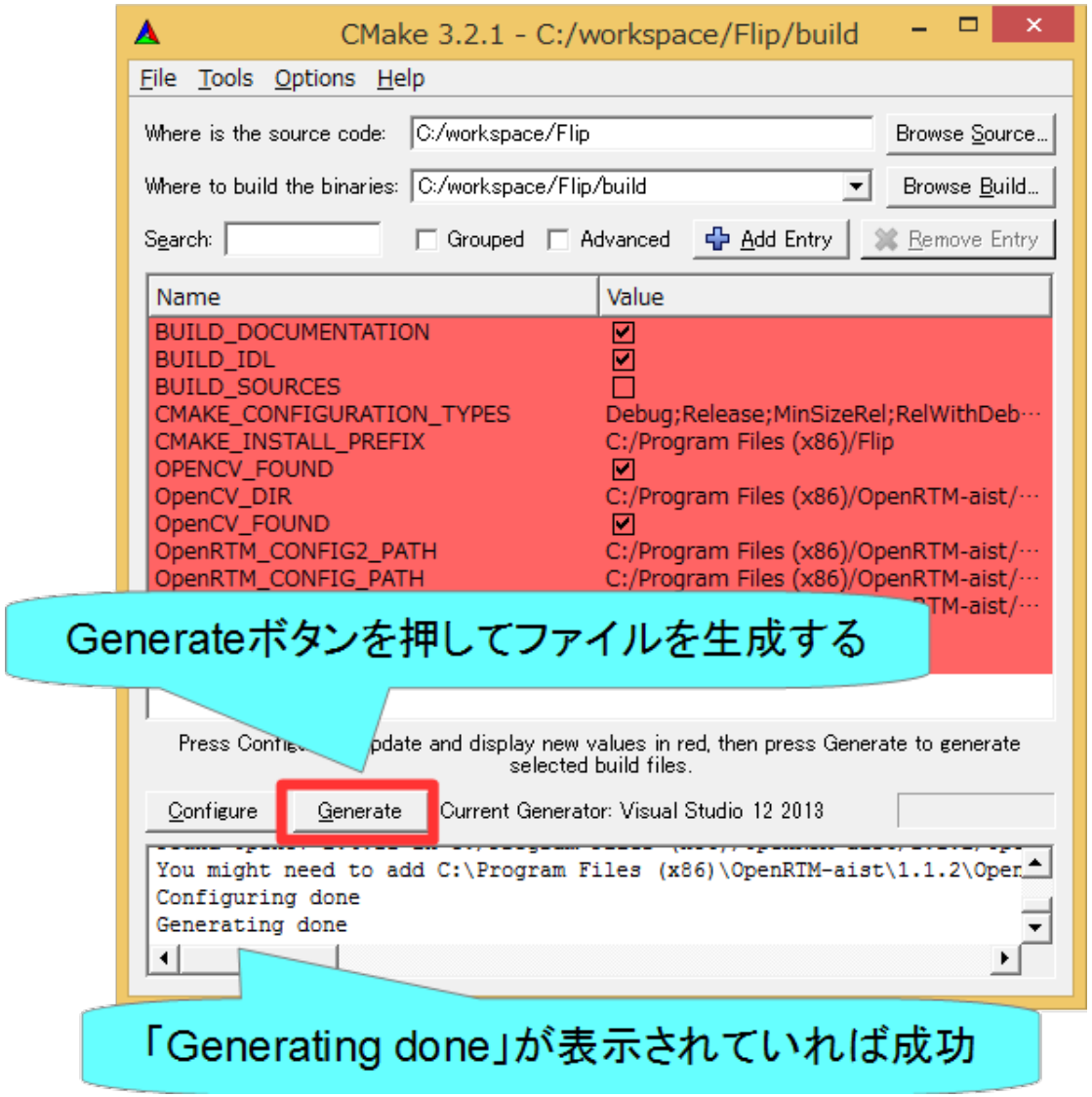


設定後、Finishボタンを押す



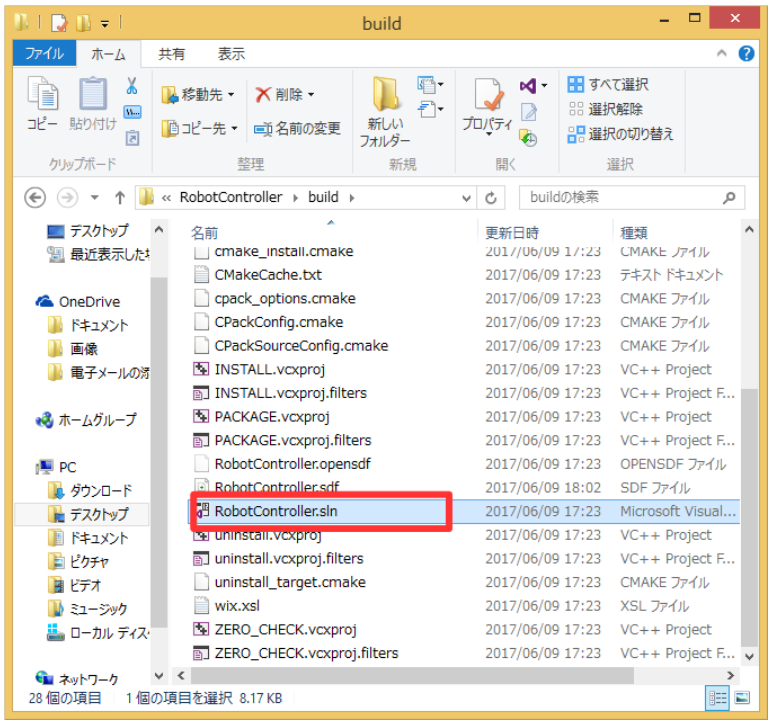
「Configure done」が表示されていれば成功

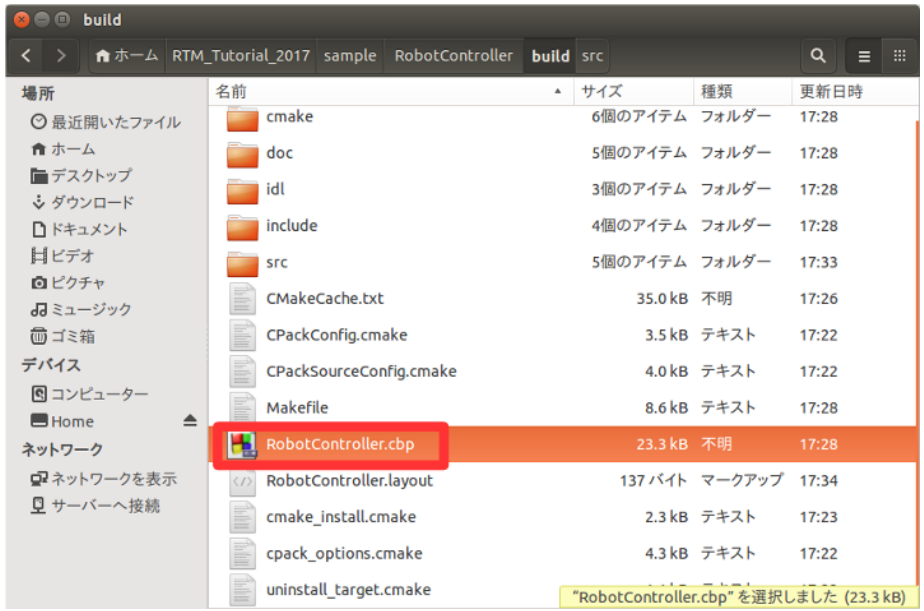
Generate files needed for build



Editing source code

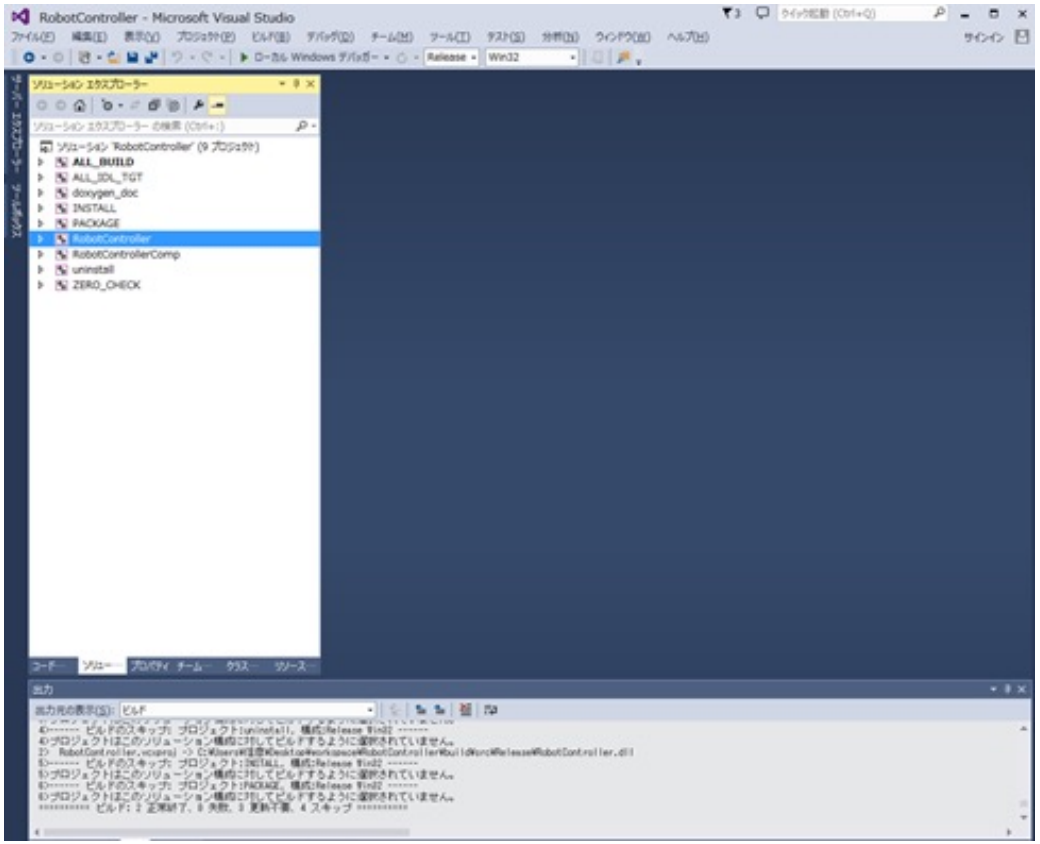
- If the version of CMake-gui is old, there is no "Open Project" button, so double-click the file to open it.
  - Windows
    - le-click "RobotController.sln" in the build folder to open it
  - Ubuntu
    - Double-click "RobotController.cbp" in the build folder to open it.

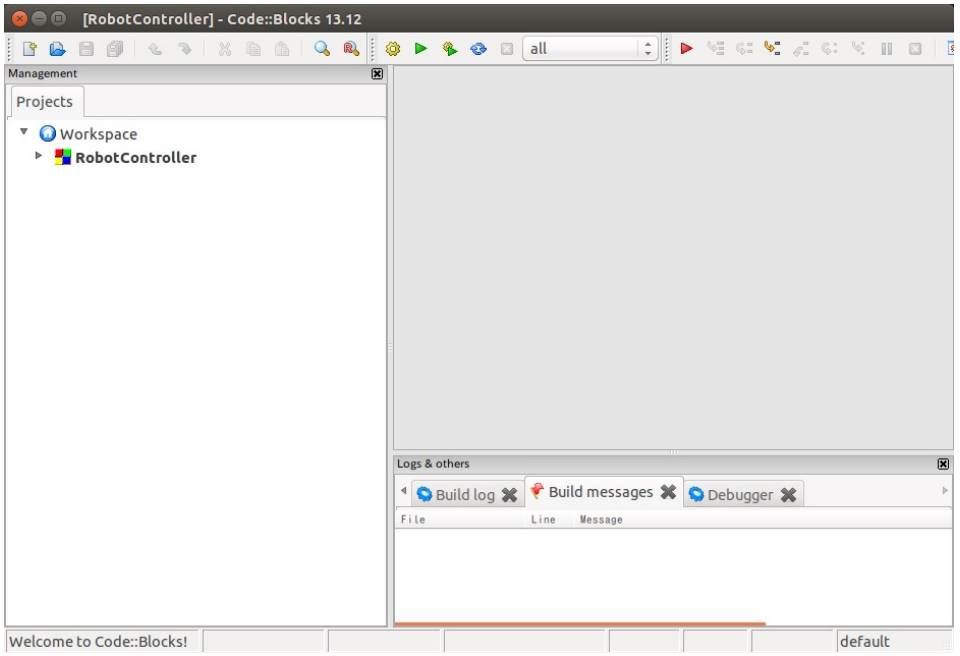




Editing source code

- Windows
  - Visual Studio starts
- Ubuntu
  - Code::Blocks starts

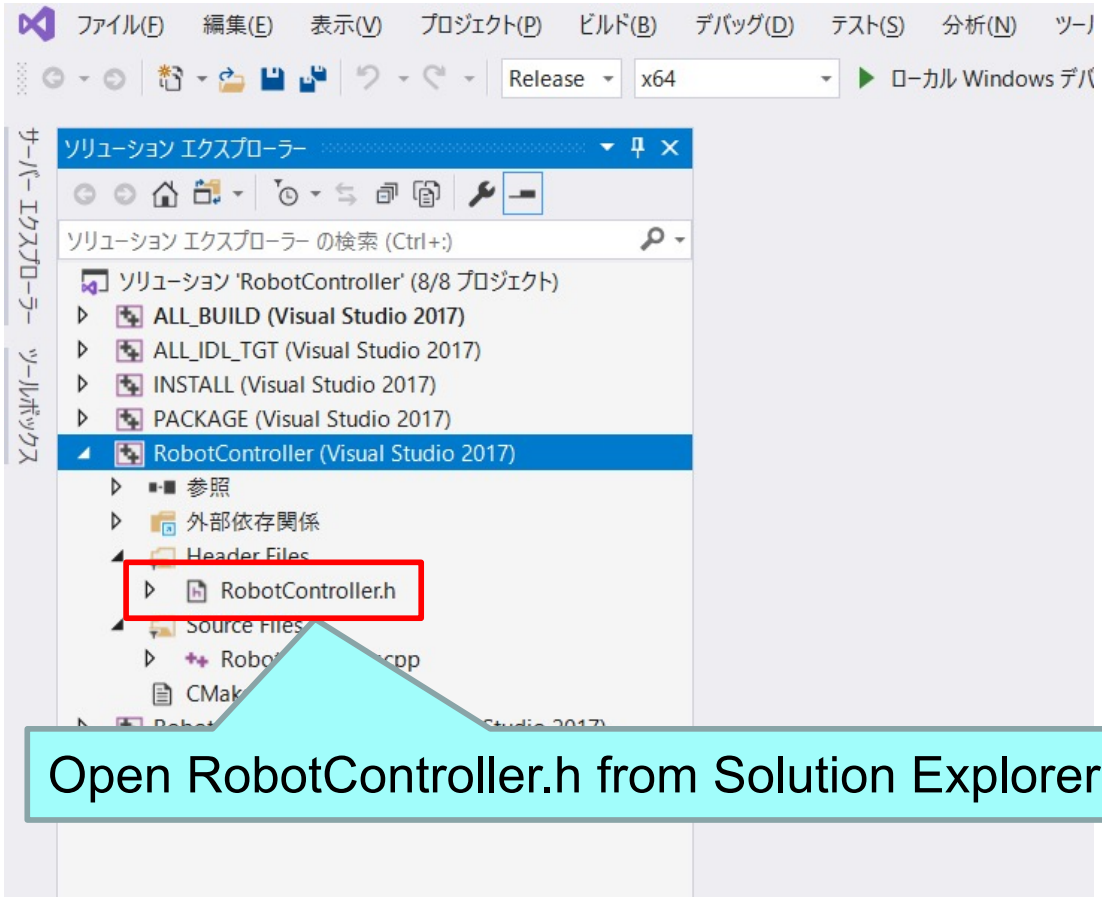




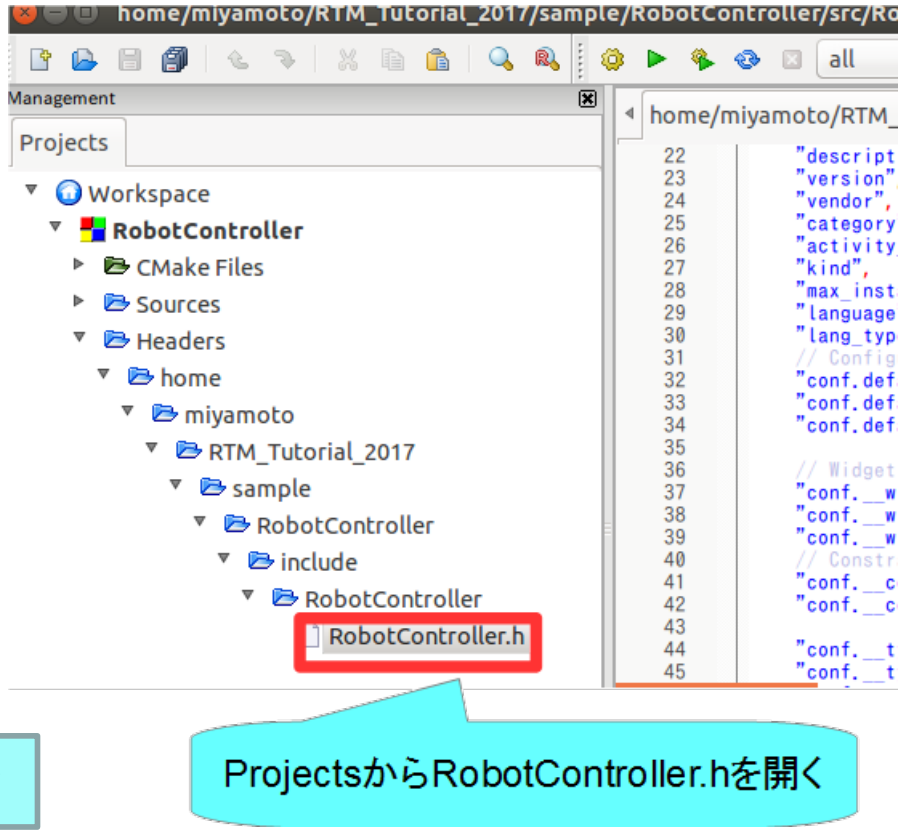
Editing source code

- Edit RobotController.h

Visual Studio



Code::Blocks



Edit source code

- Edit Flip.h

```
37  #include <rtm/CorbaPort.h>
38  #include <rtm/DataInPort.h>
39  #include <rtm/DataOutPort.h>
40
41  //OpenCV用インクルードファイルのインクルード
42  #include <opencv2/opencv.hpp>
43
44  using namespace RTC;
```

OpenCVのヘッダーファイルをインクルードする
#include <opencv2/opencv.hpp>

```
272  private:
273    // <rtc-template block="private_attribute">
274
275    // </rtc-template>
276
277    // <rtc-template block="private_operation">
278
279    // </rtc-template>
280      cv::Mat m_imageBuff;
281      cv::Mat m_flipImageBuff;
282
283  };
```

変数の宣言
cv::Mat m_imageBuff;
cv::Mat m_flipImageBuff;

Edit source code

- Edit Flip.cpp

```
110 RTC::ReturnCode_t Flip::onActivated(RTC::UniqueId ec_id)
111 {
112
113     // OutPortの画面サイズの初期化
114     m_flippedImage.width = 0;
115     m_flippedImage.height = 0;
116
117     return RTC::RTC_OK;
118 }
```

onActivateに追加

```
121 RTC::ReturnCode_t Flip::onDeactivated(RTC::UniqueId ec_id)
122 {
123     if (!m_imageBuff.empty())
124     {
125         // 画像用メモリの解放
126         m_imageBuff.release();
127         m_flipImageBuff.release();
128     }
129
130     return RTC::RTC_OK;
131 }
```

onDeactivateに追加

Edit source code

- Edit .cpp Flip

```
134 RTC::ReturnCode_t Flip::onExecute(RTC::UniqueId ec_id)
135 {
136     // 新しいデータのチェック
137     if (m_originalImageIn.isNew()) {
138         // InPortデータの読み込み
139         m_originalImageIn.read();
140 
141         // InPortとOutPortの画面サイズ処理およびイメージ用メモリの確保
142         if (m_originalImage.width != m_flippedImage.width || m_originalImage.height != m_flippedImage.height)
143         {
144             m_flippedImage.width = m_originalImage.width;
145             m_flippedImage.height = m_originalImage.height;
146 
147             m_imageBuff.create(cv::Size(m_originalImage.width, m_originalImage.height), CV_8UC3);
148             m_flipImageBuff.create(cv::Size(m_originalImage.width, m_originalImage.height), CV_8UC3);
149 
150 
151         }
152 
153         // InPortの画像データをm_imageBuffにコピー
154         memcpy(m_imageBuff.data, (void *)&(m_originalImage.pixels[0]), m_originalImage.pixels.length());
155 
156         // InPortからの画像データを反転する。 m_flipMode 0: X軸周り, 1: Y軸周り, -1: 両方の軸周り
157         cv::flip(m_imageBuff, m_flipImageBuff, m_flipMode);
158 
159         // 画像データのサイズ取得
160         int len = m_flipImageBuff.channels() * m_flipImageBuff.cols * m_flipImageBuff.rows;
161         m_flippedImage.pixels.length(len);
162 
163         // 反転した画像データをOutPortにコピー
164         memcpy((void *)&(m_flippedImage.pixels[0]), m_flipImageBuff.data, len);
165 
166         // 反転した画像データをOutPortから出力する。
167         m_flippedImageOut.write();
168     }
169 
170     return RTC::RTC_OK;
171 }
```

onExecuteに追加

Edit source code

- Steps to load data
-

isNew関数で新規に書き込まれたデータが存在するかを確認

```
// 新しいデータのチェック
if (m_originalImageIn.isNew()) {
    // InPortデータの読み込み
    m_originalImageIn.read();

    // InPortとOutPortの画面サイズ処理およびイメージ用メモリの確保
    if (m_originalImage.width != m_flippedImage.width || m_originalImage.height != m_flippedImage.height)
    {
```

read関数でデータの読み込み

read関数を呼び出した時点で
変数m_originalImageにデータが格納される

- Steps to write data
-

変数m_flippedImageにデータを格納

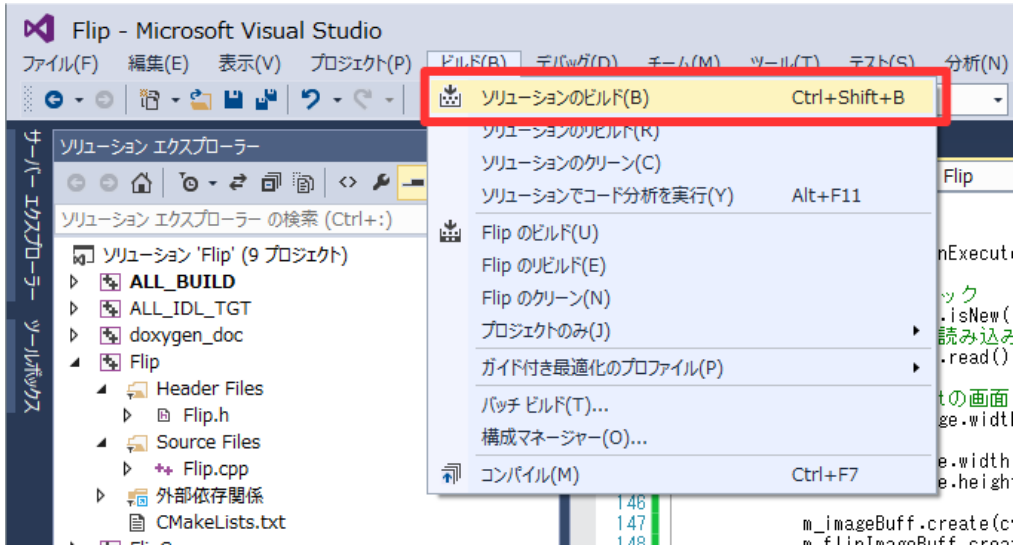
```
// 反転した画像データをOutPortにコピー
memcpy((void *)&(m_flippedImage.pixels[0]), m_flipImageBuff.data, len);

// 反転した画像データをOutPortから出力する。
m_flippedImageOut.write();
```

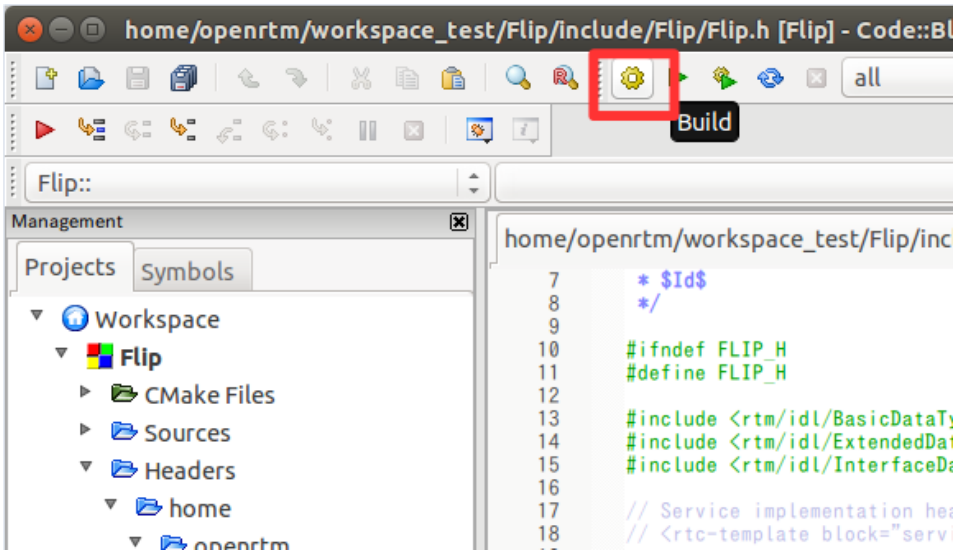
write関数でデータの書き込み

Compiling source code

Visual Studio



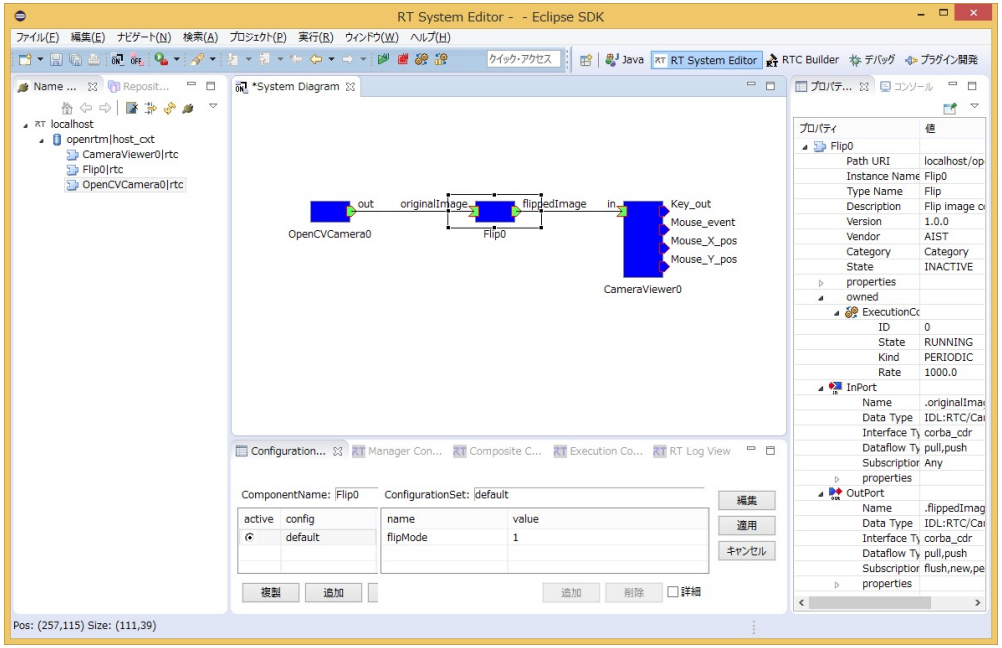
Code::Blocks



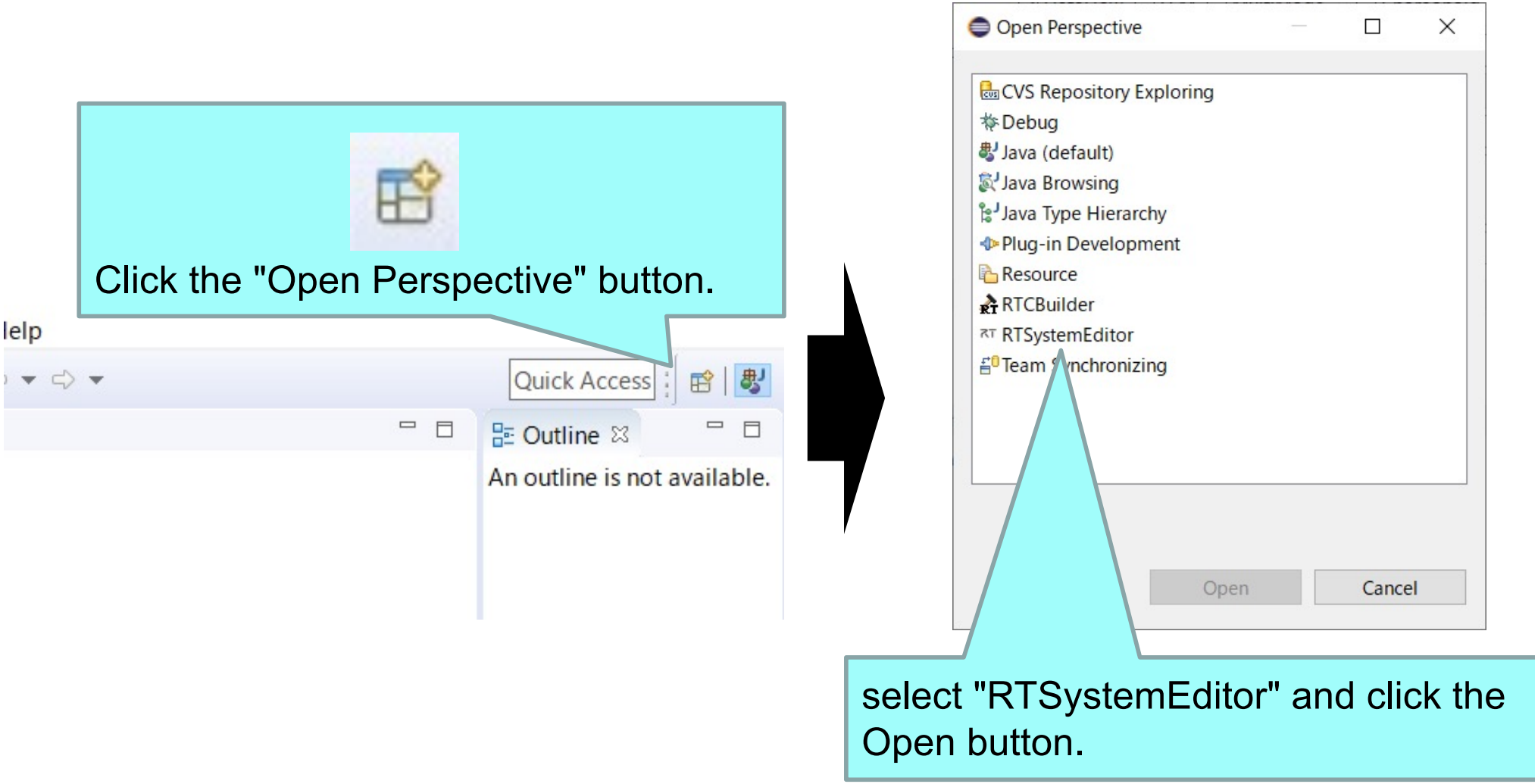
System construction support tool
RT System Editor

RT System Editor

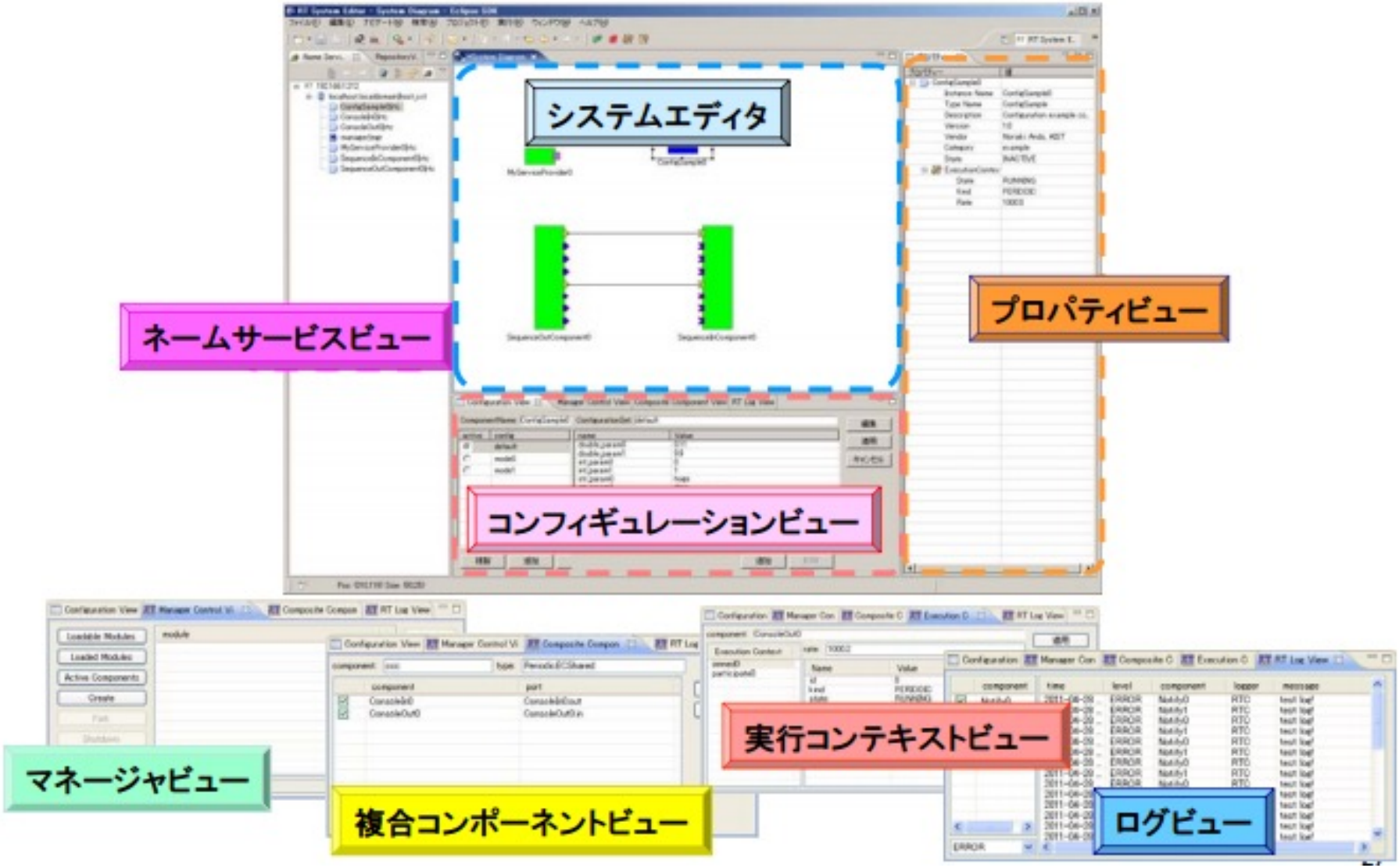
- Tool for operating RTC with GUI
  - Data port and service port connection
  - Activate, deactivate, reset, exit
  - Manipulating configuration parameters
  - Manipulating the execution context
    - Change execution cycle
    - Execution context association
  - Composite
  - Launch RTC from manager
  - Save and restore the created RT system



Start RTSystemEditor

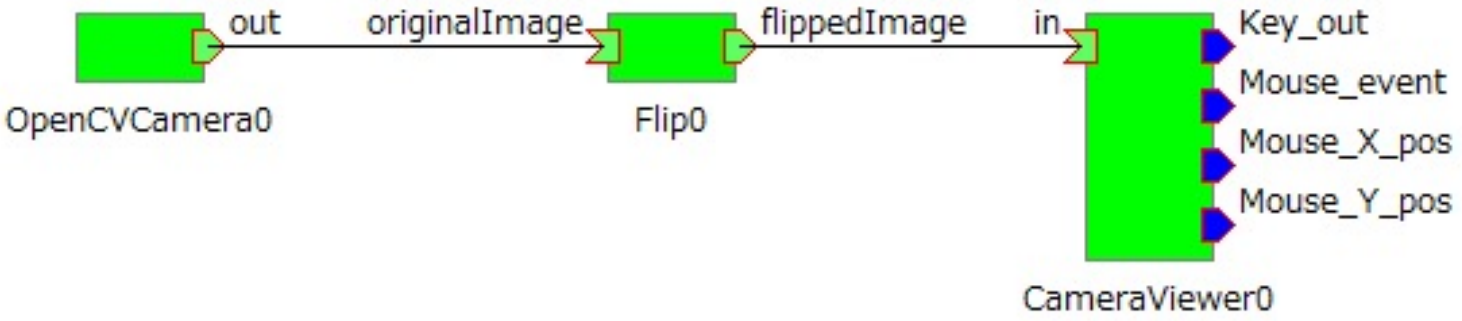


RT System Editor screen configuration



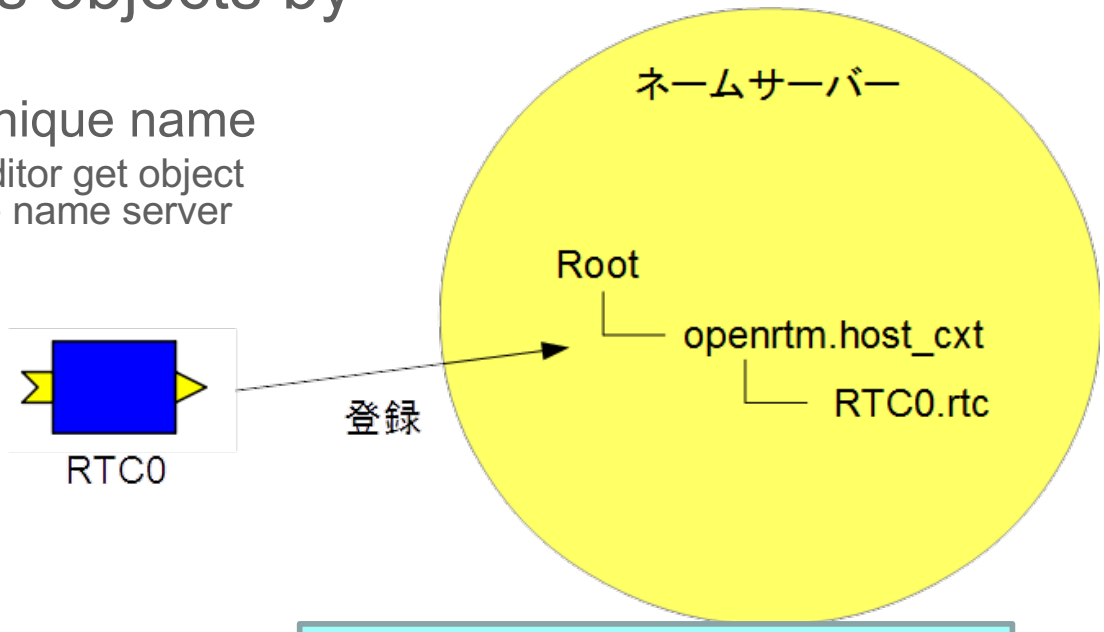
Check the operation of flip components

- Create an RT system that inverts and displays images taken with a webcam
  - Start a name server
  - Launch CameraViewer RTC and OpenCVCamera RTC
    - Windows
      - 「OpenRTM-aist 1.1.2」→「C++」→「Components」→「OpenCVExamples」
    - Ubuntu
      - $ /usr/local/share/openrtm-1.1/components/c++/opencv-rtcs/CameraViewerComp
      - $ /usr/local/share/openrtm-1.1/components/c++/opencv-rtcs/OpenCVCameraComp
  - Launch Flip RTC
    - Windows
      - FlipComp.exe is generated under **Release** or **Debug** folder in **build¥src folder**, and launch it
    - Ubuntu
      - FlipComp is generated in **build/src** folder, and launch it.
  - Connecting it with CameraViewer RTC and OpenCVCamera RTC, and “All Activate” the system.

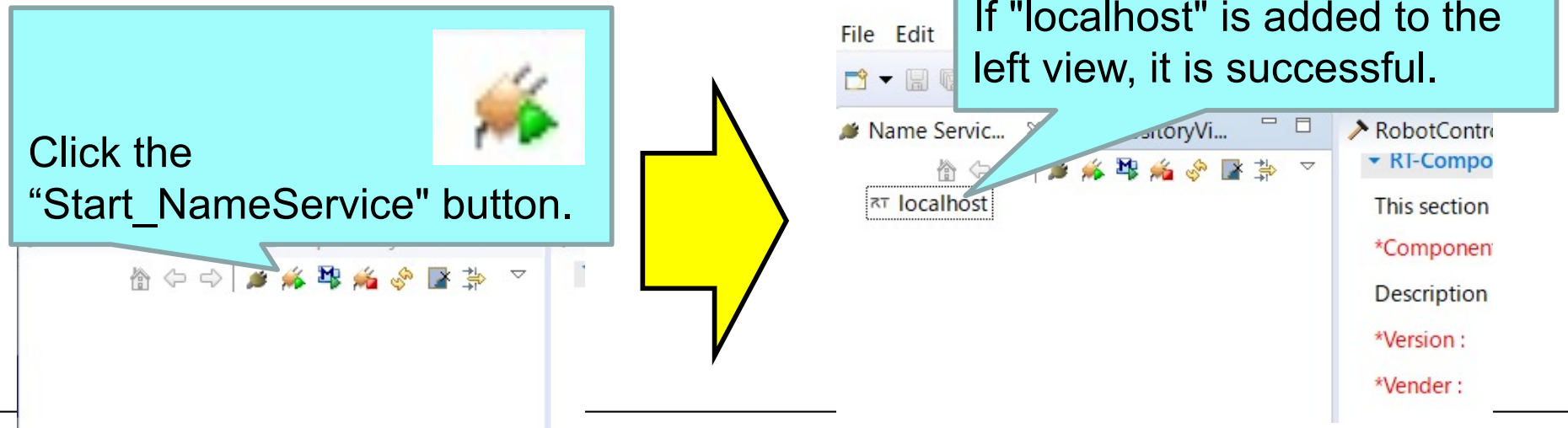


Naming Service

- A service that manages objects by name
  - Register the RTC with a unique name
    - Tools such as RT System Editor get object references by name from the name server



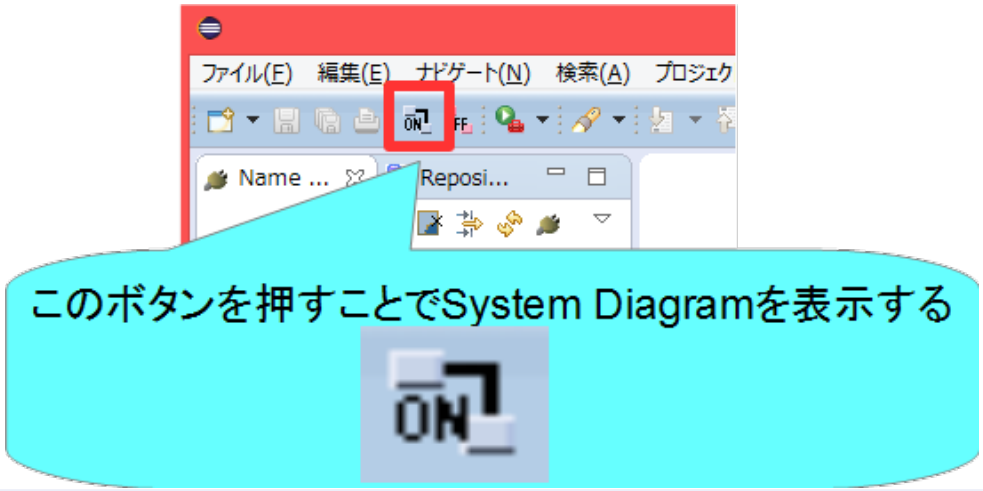
- Procedure to start

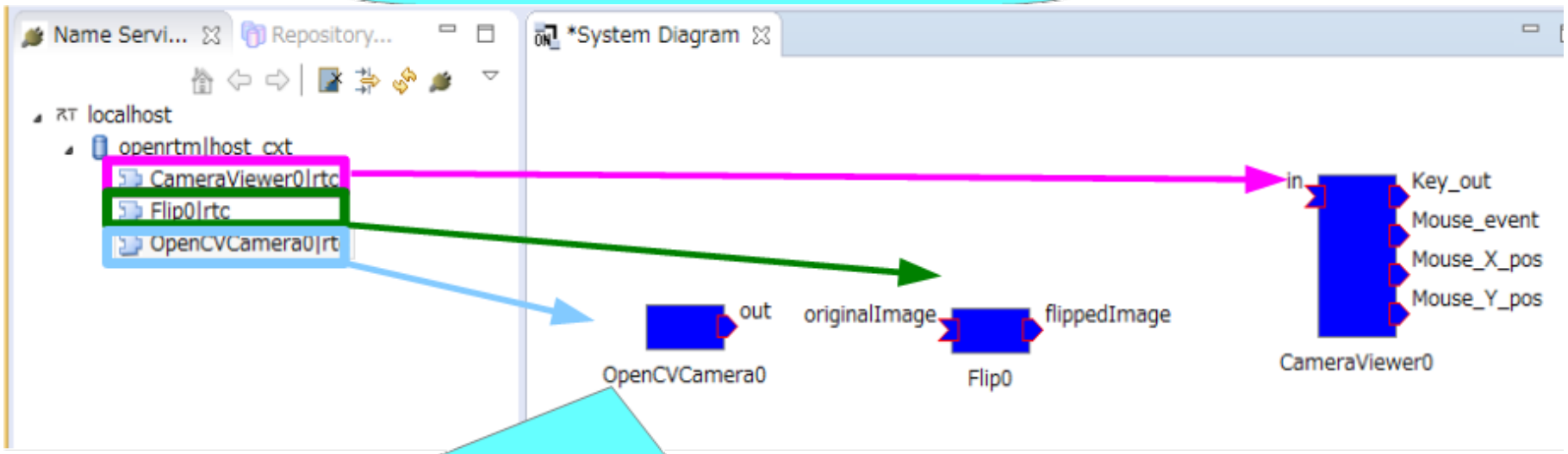


Start Naming Service

- OpenRTM-aist 1.1.2以前の手順
  - Windows 7
    - 「スタート」→「すべてのプログラム」→「OpenRTM-aist 1.2.0」→「Tools」→「Start Naming Service」
  - Windows 8.1
    - 「スタート」→「アプリビュー(右下矢印)」→「OpenRTM-aist 1.2.0」→「Start Naming Service」
  - Windows 10
    - 左下の「ここに入力して検索」にStart Naming Serviceと入力して起動
  - Ubuntu
    - $ rtm-naming

Connecting data ports



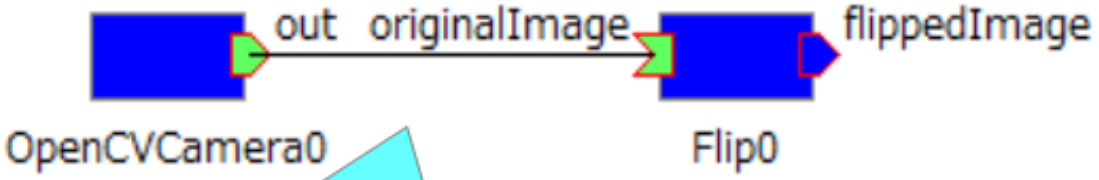


左側のネームサービスビューから
CameraViewer0.rtc、OpenCVCamera0.rtc、Flip0.rtcを
System Diagramにドラッグアンドドロップ

Connecting data ports

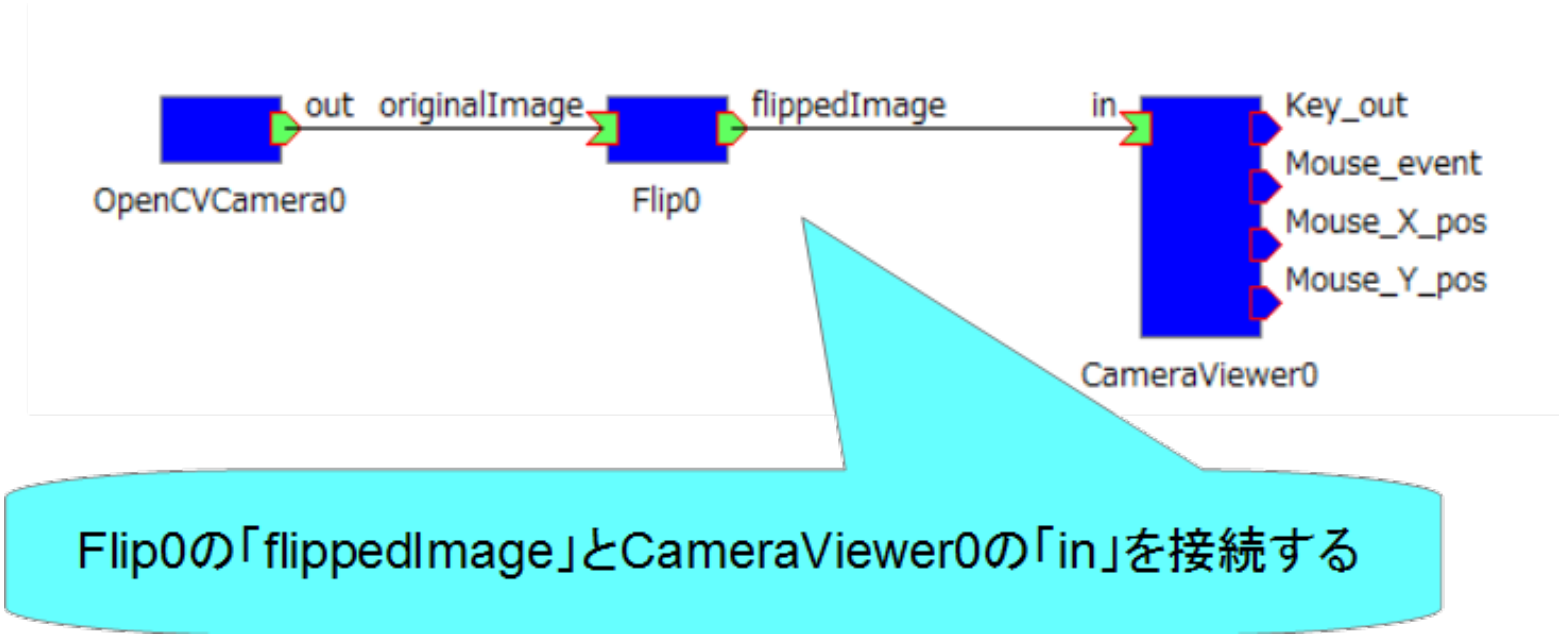
OpenCVCamera0の「out」を選択して、
Flip0の「originalImage」にドラッグアンドドロップ

そのままの設定でOKをクリック

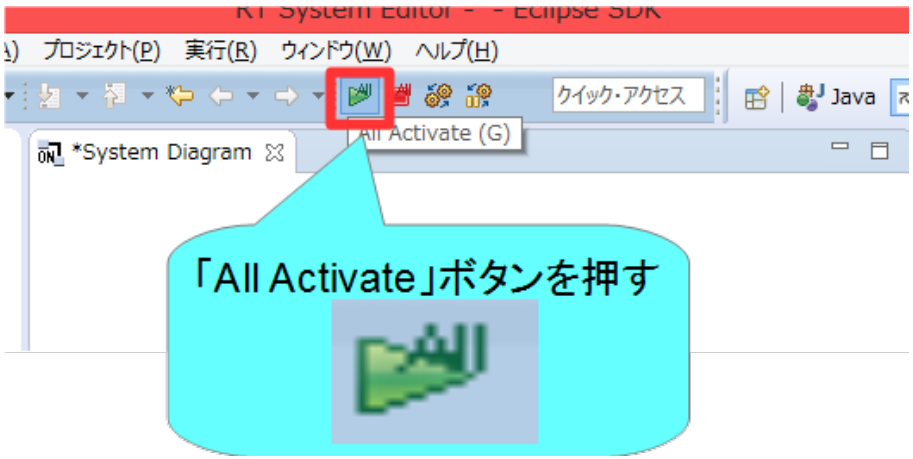


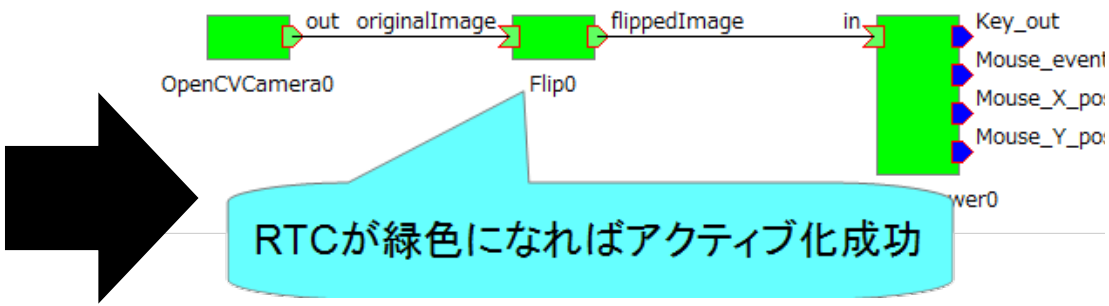
1. ポートの間に線が表示される
2. InPort、OutPortが緑色で表示される

Connecting data ports



Activate

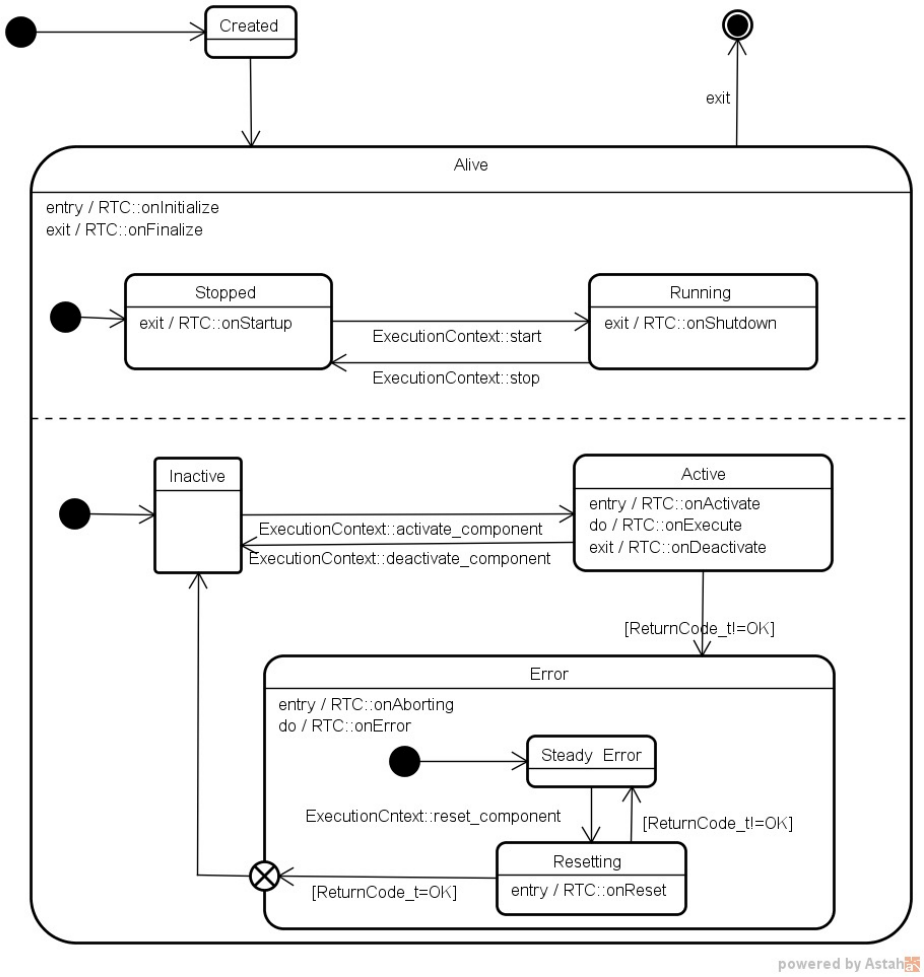




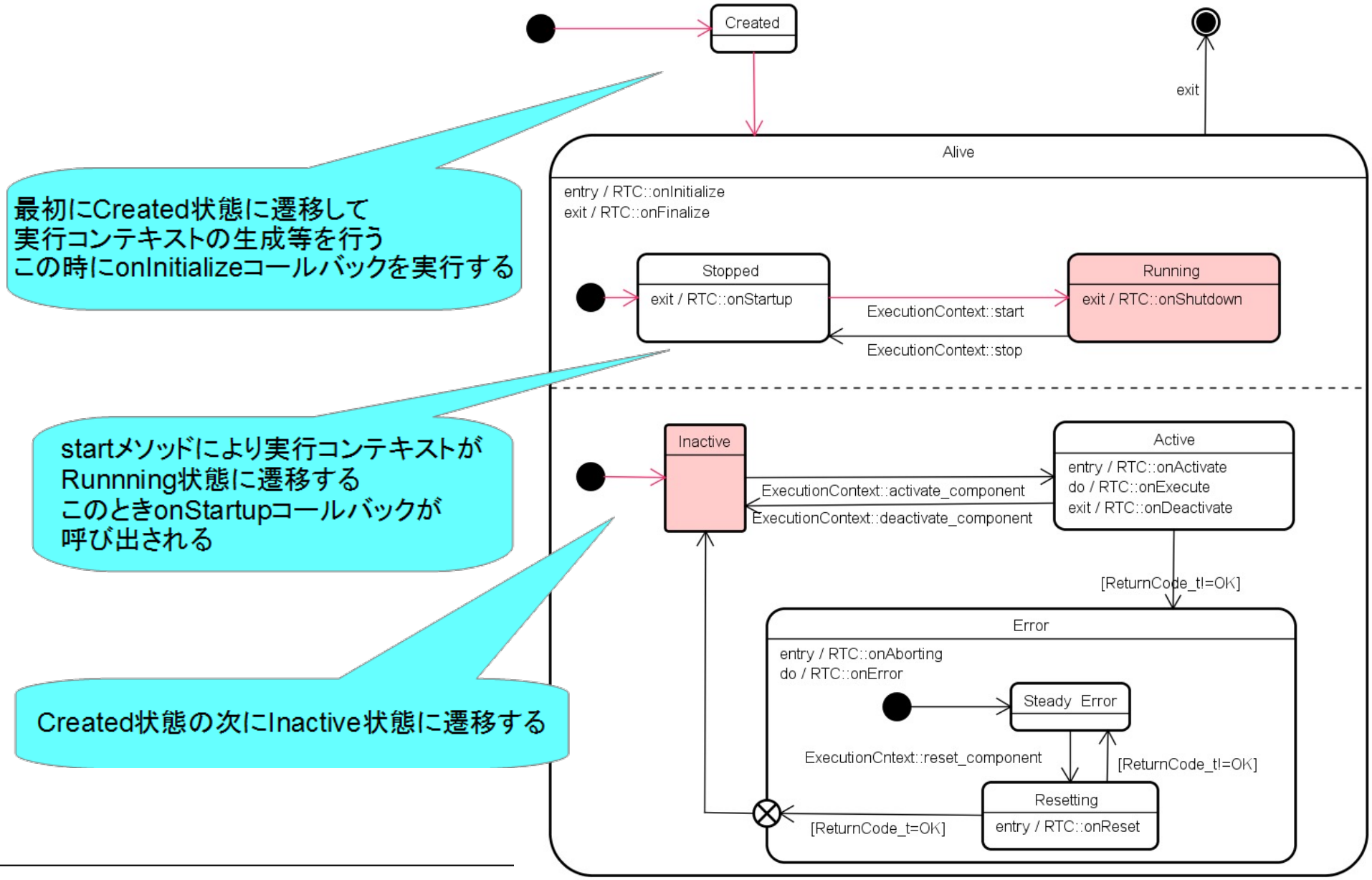
- Check if the image taken with the webcam is displayed inverted
  - If you don't see it
    - The camera is not connected to the PC
    - Data port not connected
    - RTC is not active

RT-Component state transitions

- RTC has the following conditions:
  - Created
    - Generates an execution context and start() is called to put the thread in the execution context in a running state
    - Automatically transitions to the Inactive state
  - Inactive
    - activate_component method to transition to an active state
    - Display on RT System Editor is blue
  - Active
    - OnExecute callback is executed by execution context
    - The return code RTC_OK to an error state other than the return code
    - Green display on RT System Editor
  - Error
    - OnErrrer callback is executed by execution context
    - reset_component methods to transition to an inactive state
    - Display on RT System Editor is red
  - End state



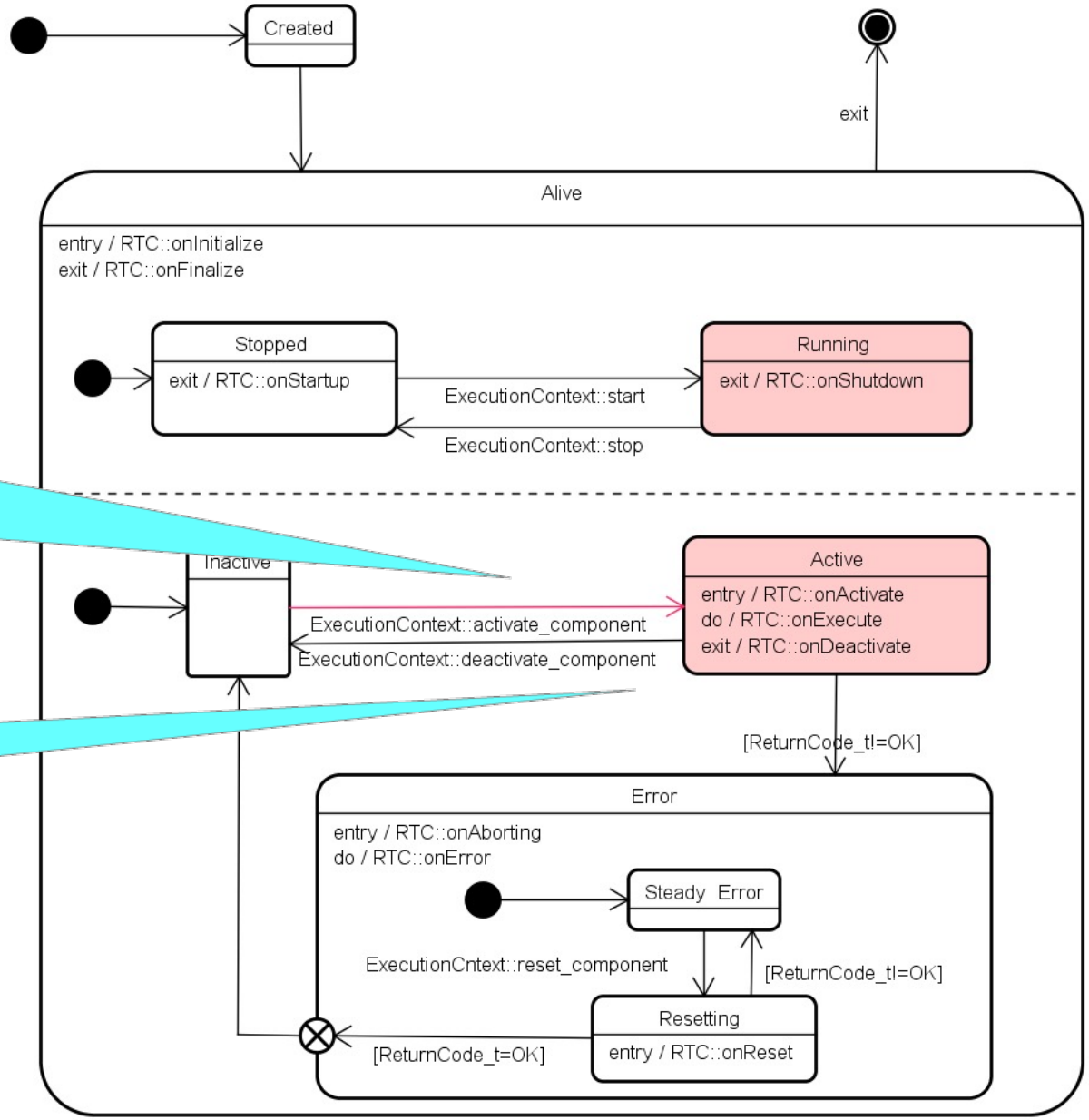
RT component state transition (immediately after generation)



RT component state transition (activation)

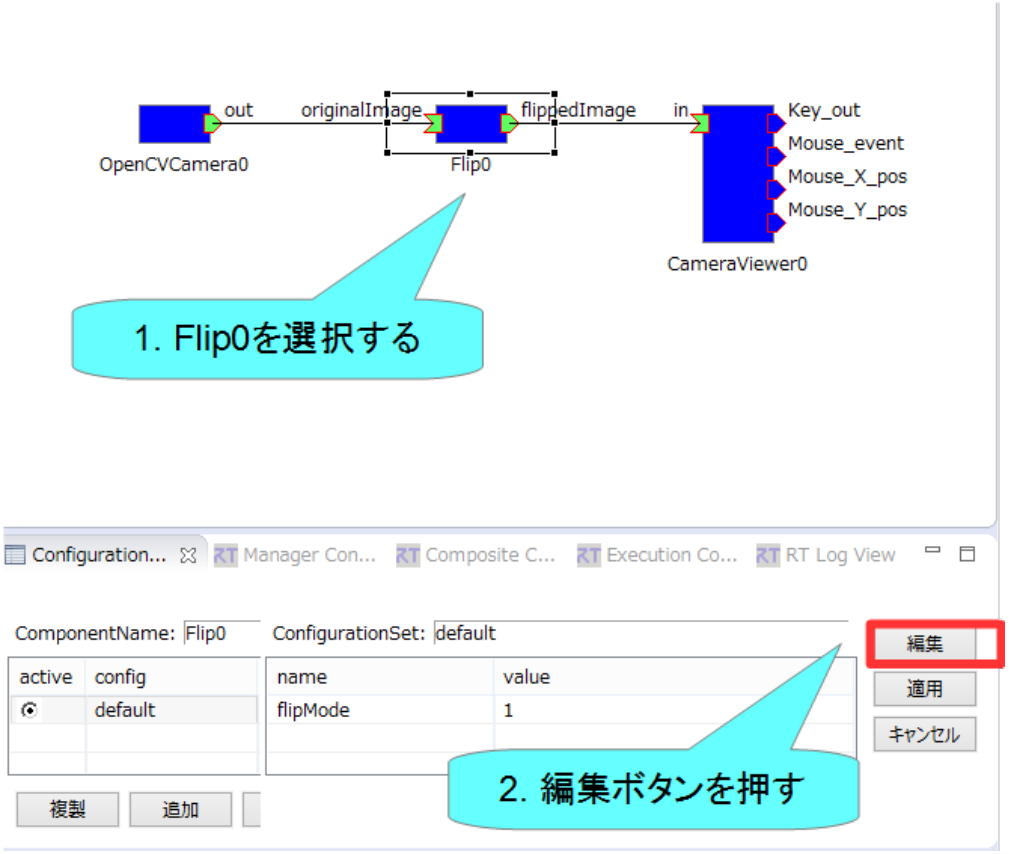
RTシステムエディタの操作によりRTコンポーネントのアクティブ化を行うとactivate_componentメソッドが呼び出される。
activate_componentメソッドによりコンポーネントがActive状態に遷移する。
この時onActivatedコールバックが実行される

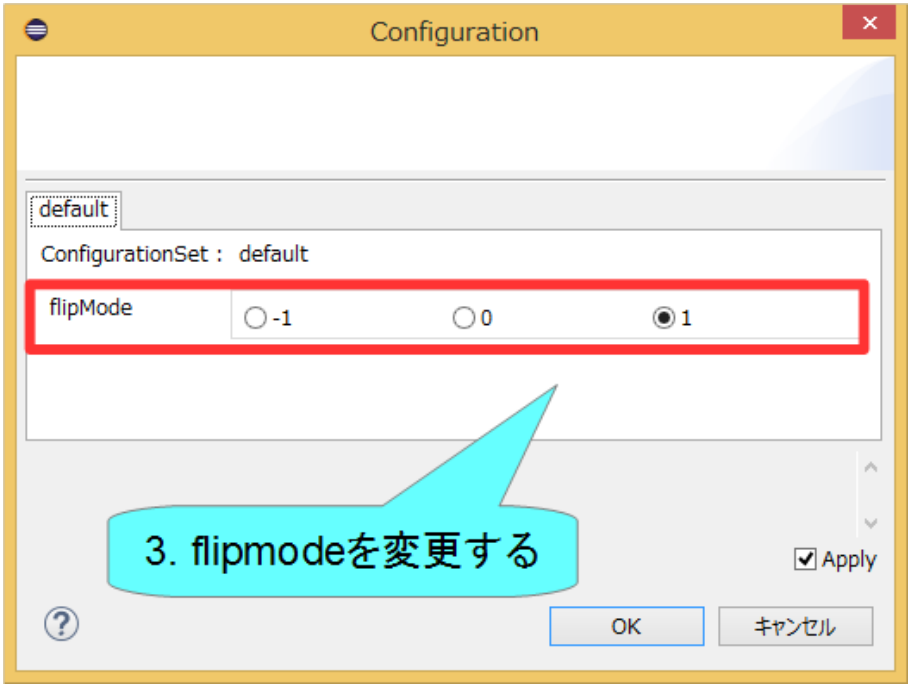
周期実行の実行コンテキストの場合、
onExecuteコールバックが周期的に呼び出される。



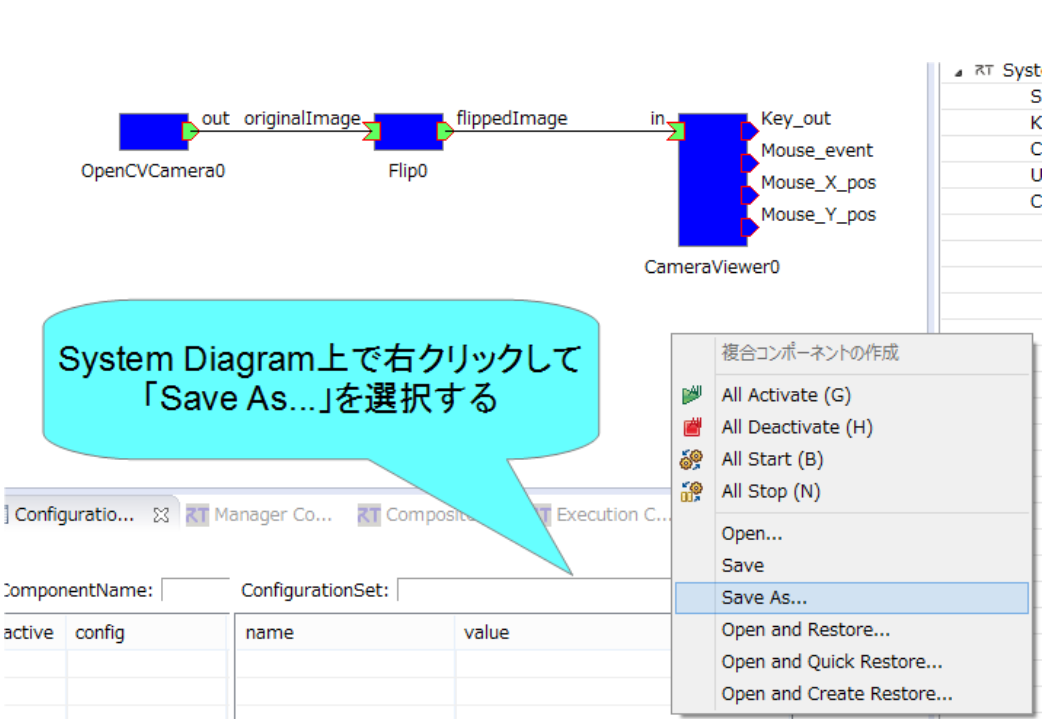
Working with Configuration Parameters

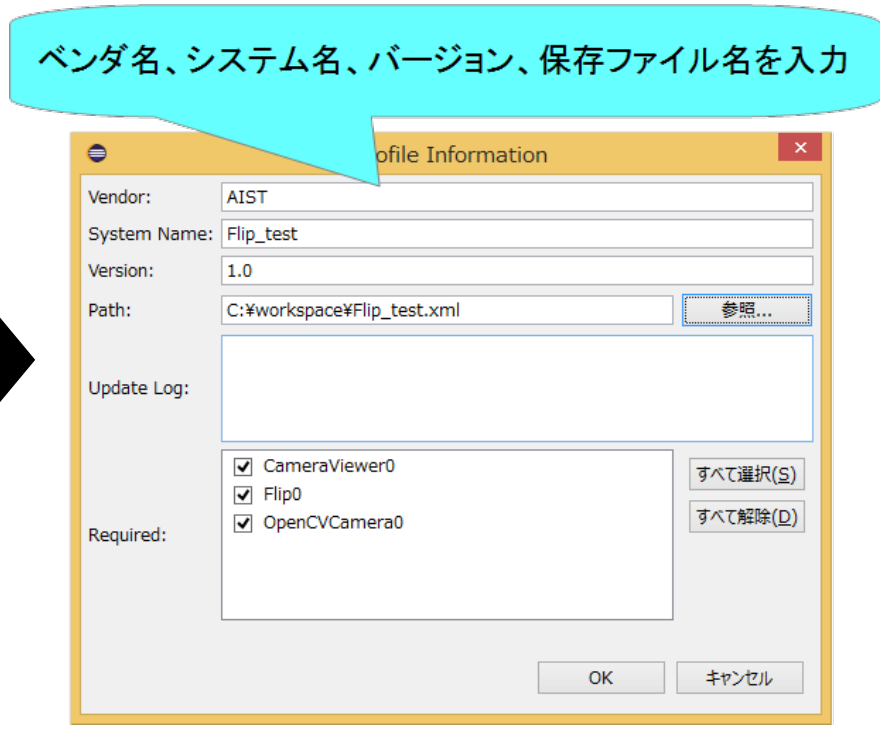
- Working with configuration parameters from the RT system editor
  - Set the direction to flip



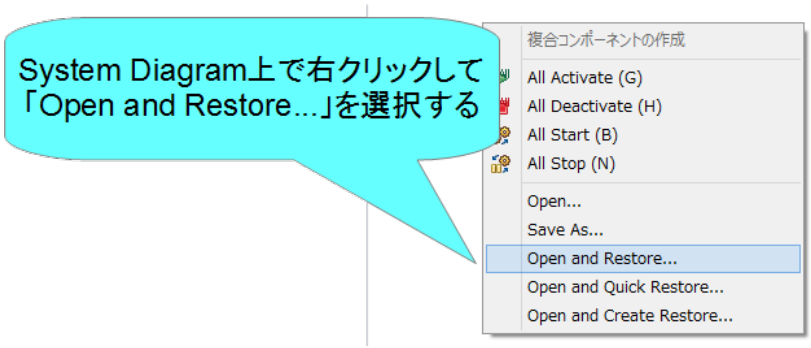


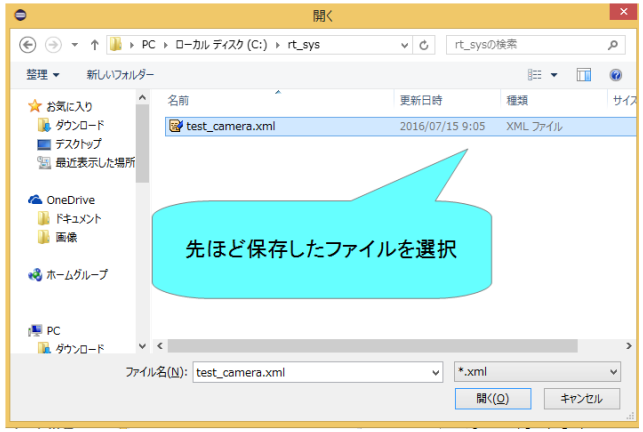
Save system

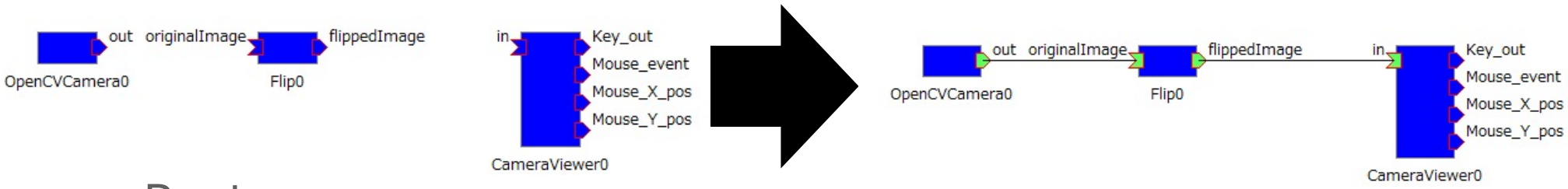

out
originalImage
flippedImage
in
Key_out
Mouse_event
Mouse_X_pos
Mouse_Y_pos
OpenCVCamera0
Flip0
CameraViewer0
System Diagram上で右クリックして
「Save As...」を選択する
複合コンポーネントの作成
All Activate (G)
All Deactivate (H)
All Start (B)
All Stop (N)
Open...
Save
Save As...
Open and Restore...
Open and Quick Restore...
Open and Create Restore...
ComponentName:
ConfigurationSet:
active
config
name
value


ベンダ名、システム名、バージョン、保存ファイル名を入力
ofile Information
Vendor:
AIST
System Name:
Flip_test
Version:
1.0
Path:
C:¥workspace¥Flip_test.xml
参照...
Update Log:
Required:
CameraViewer0
Flip0
OpenCVCamera0
すべて選択(S)
すべて解除(D)
OK
キャンセル

Restore system





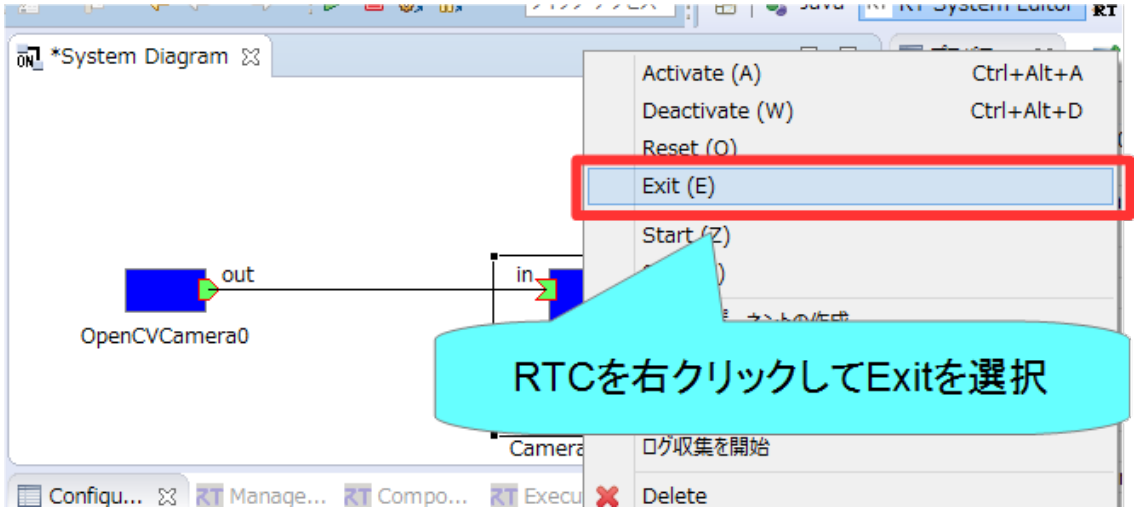


- Restore:
  - Connections between ports
  - Configuration
  - If Open and Create Restore is selected, the manager starts the component

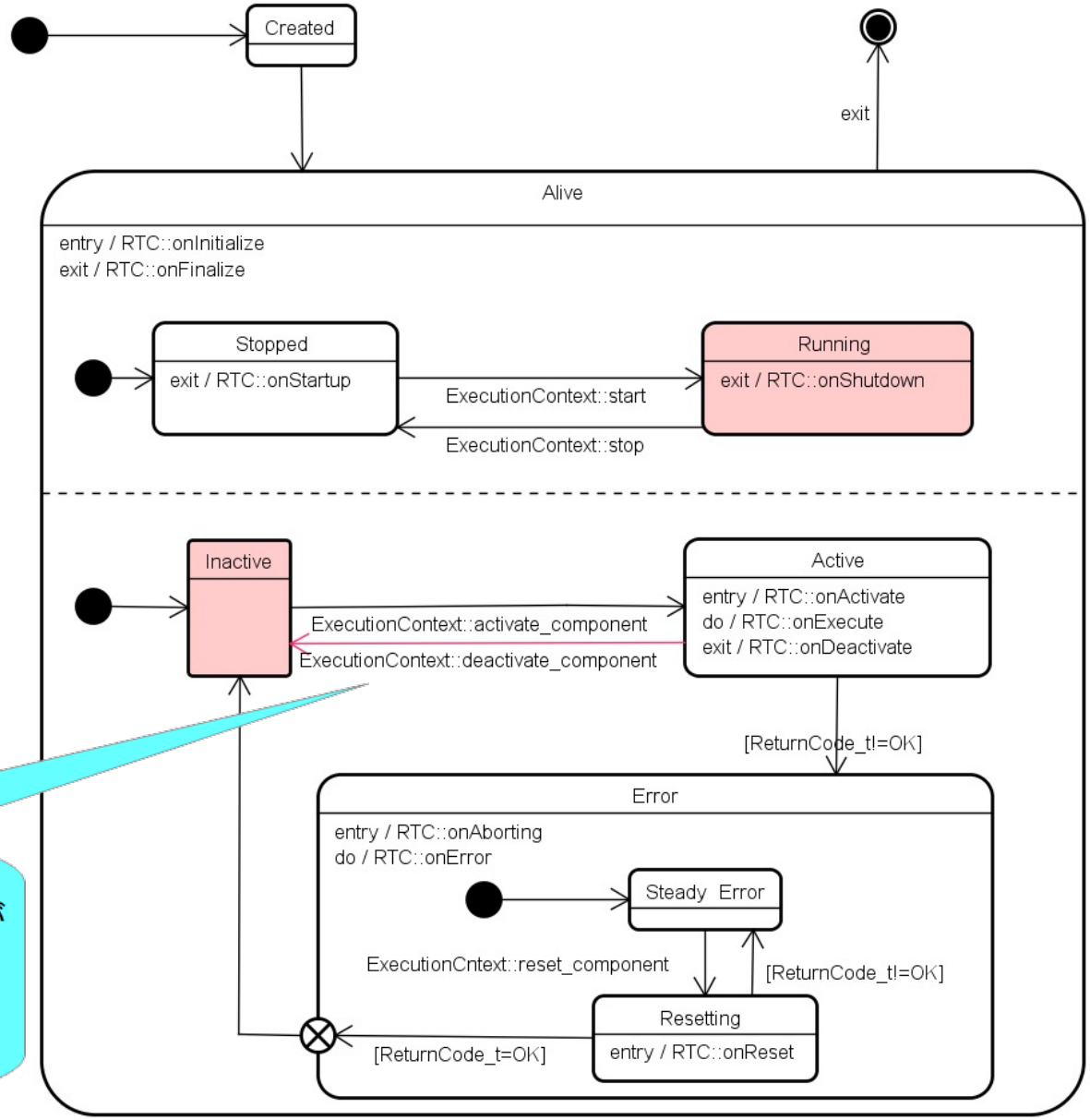
Deactivate, Exit

- Deactivation

- Eixt



RT component state transition (deactivation)

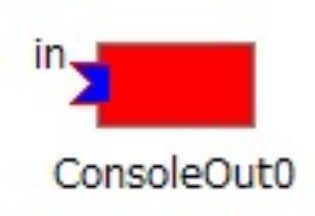


RTシステムエディタの操作によりRTコンポーネントの非アクティブ化を行うとdeactivate_componentメソッドが呼び出される。
deactivate_componentメソッドによりコンポーネントがInactive状態に遷移する。
この時onDeactivatedコールバックが実行される

RTCBuilder
Supplementary explanation

Resetting

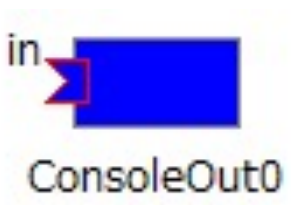
- Displayed in red on the editor when the RTC transitions to an error state。



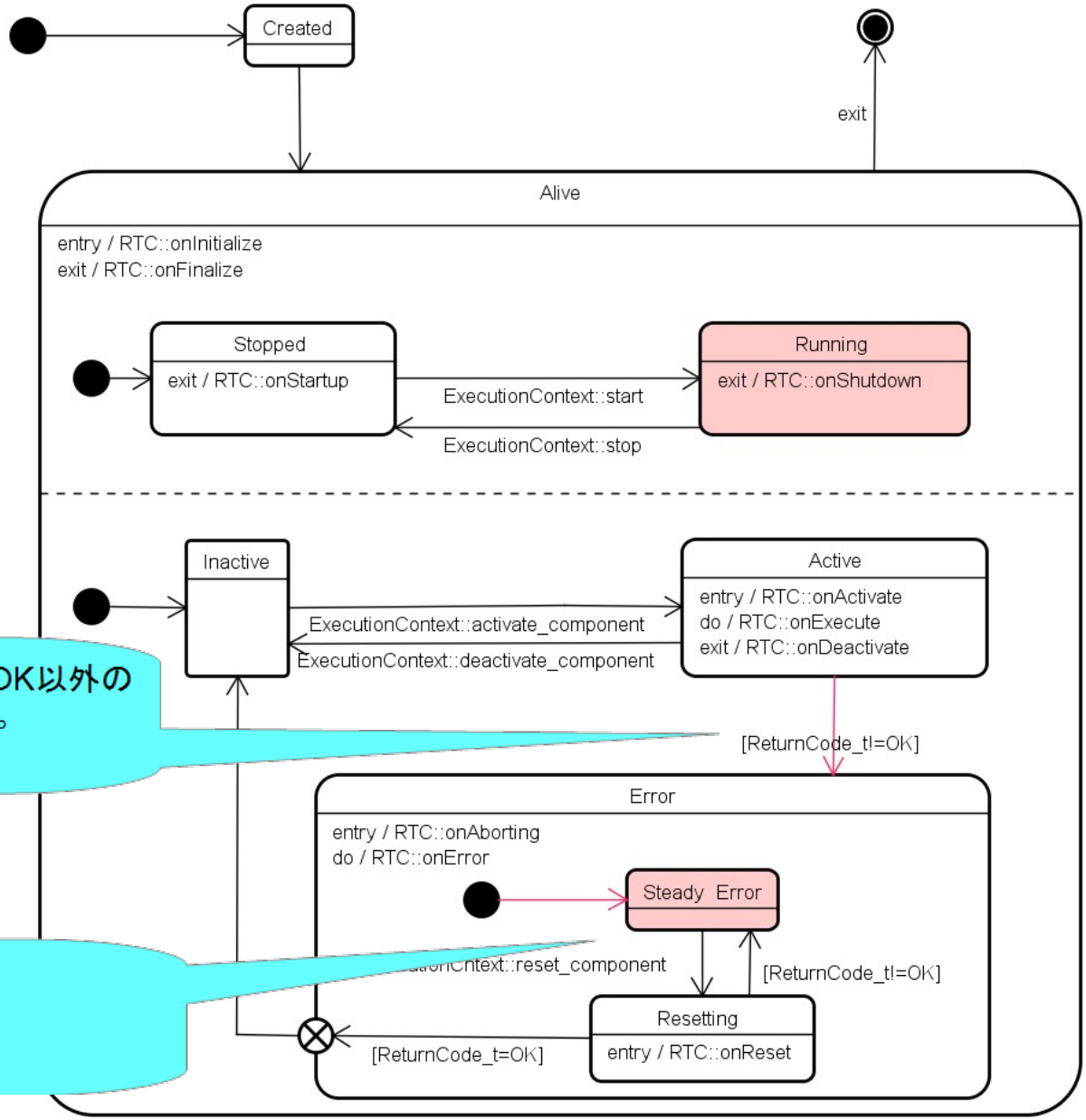

```
RTC::ReturnCode_t Test::onActivated(RTC::UniqueId ec_id)
{
        HANDLE hCom = INVALID_HANDLE_VALUE;
        hCom = CreateFile("COM5", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);
        if (hCom == INVALID_HANDLE_VALUE)
        {
                return RTC::RTC_ERROR;
        }
```

例えばonActivated関数で初期化(この例ではCOMポートの初期化)
に失敗した場合はRTC_ERRORを返すようにしておけば、
初期化に失敗した場合にエラー状態に遷移する

- Return to inactive by:
-



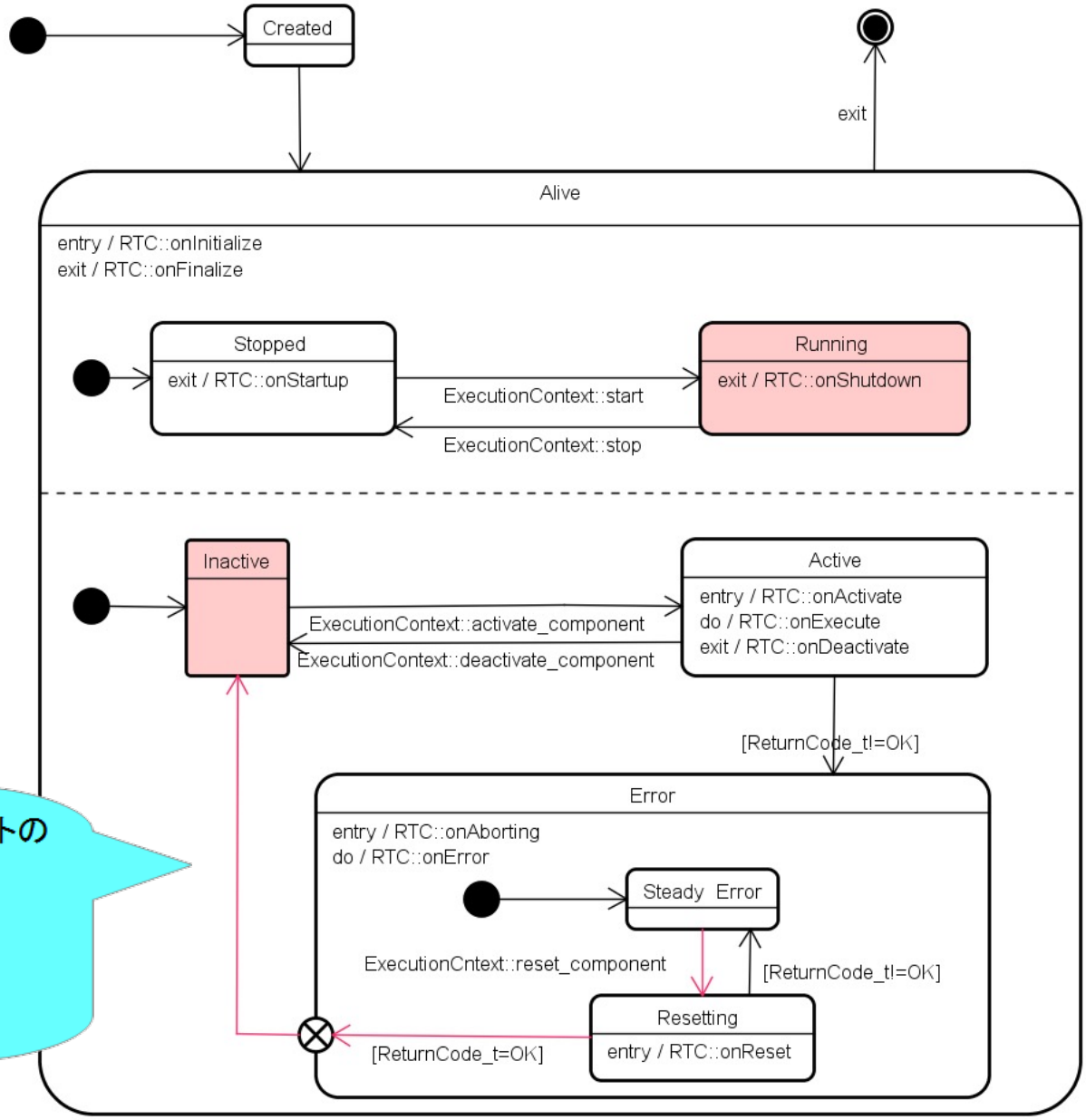
RT component state transition (error)



onActivated、onExecute、onDeactivatedがRTC_OK以外のリターンコードを返した場合、エラー状態に遷移する。
この時、onAbortingコールバックが実行される。

周期実行の実行コンテキストの場合、
onErrorコールバックが周期的に呼び出される。

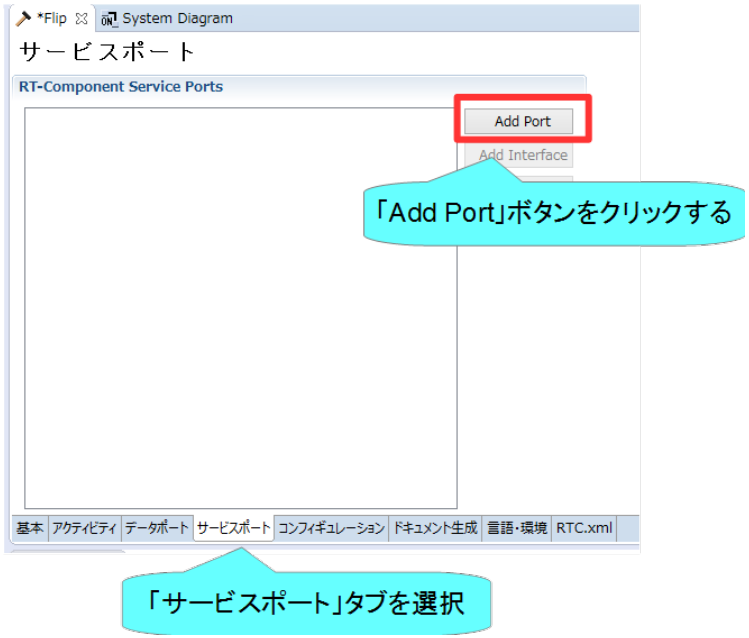
RT component state transition (reset)

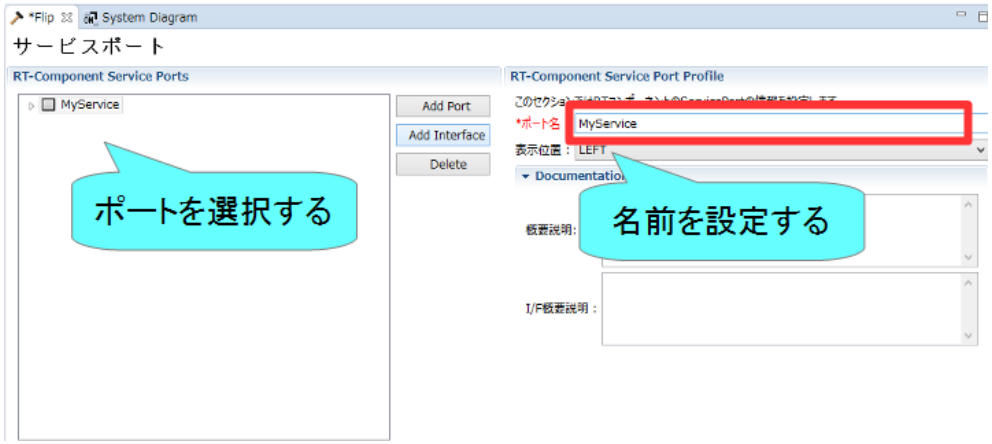


RTシステムエディタの操作によりRTコンポーネントのリセットを行うとreset_componentメソッドが呼び出される。
reset_componentメソッドによりコンポーネントがInactive状態に遷移する。
この時onResetコールバックが実行される

Set up service ports

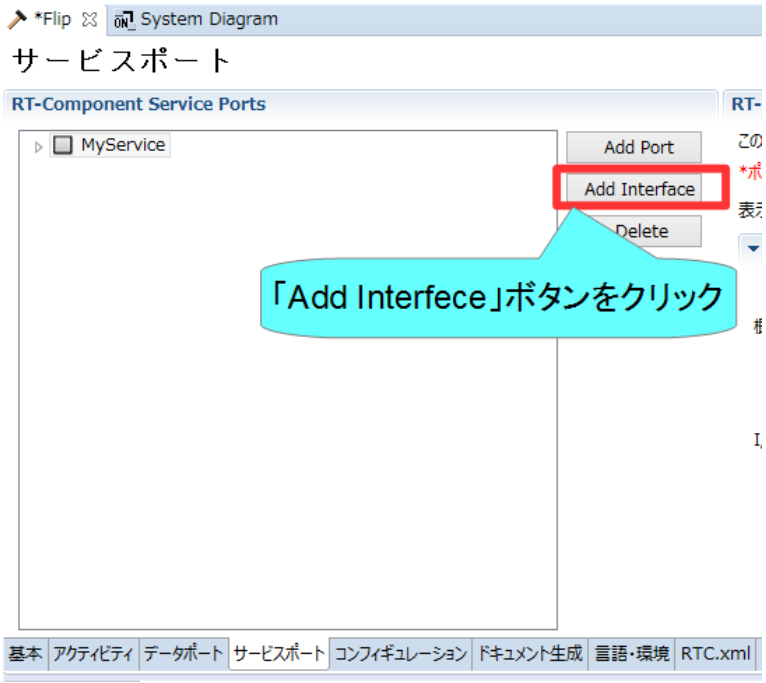
- Add service ports, add interfaces, and set up

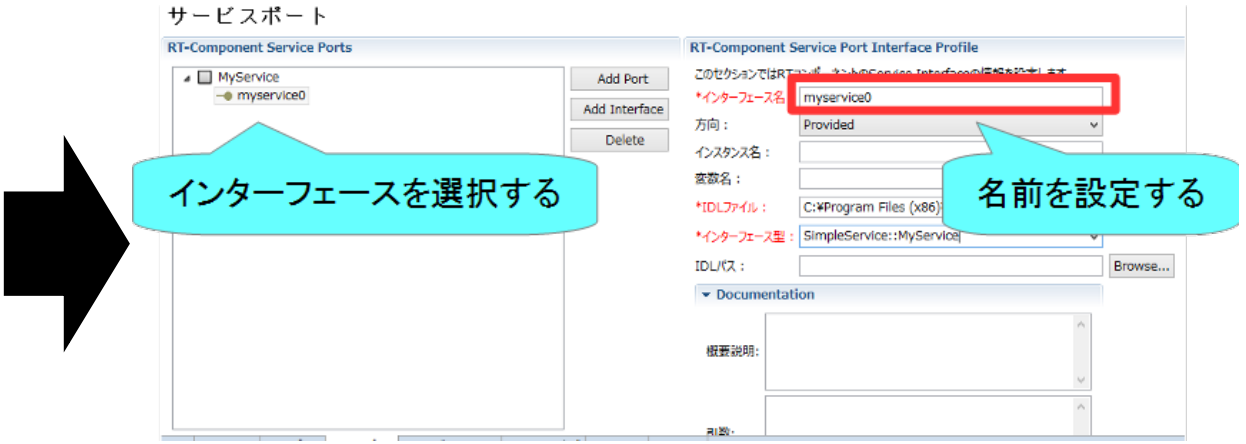




Set up service ports

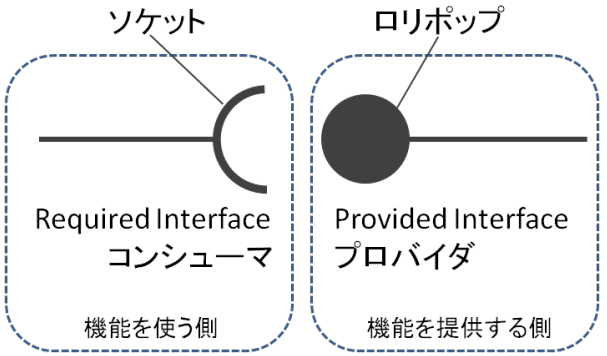
- Add an interface





Set up service ports

- Set up the interface



- After code generation, launch idlcompile .bat (idlcompile.sh) in Python

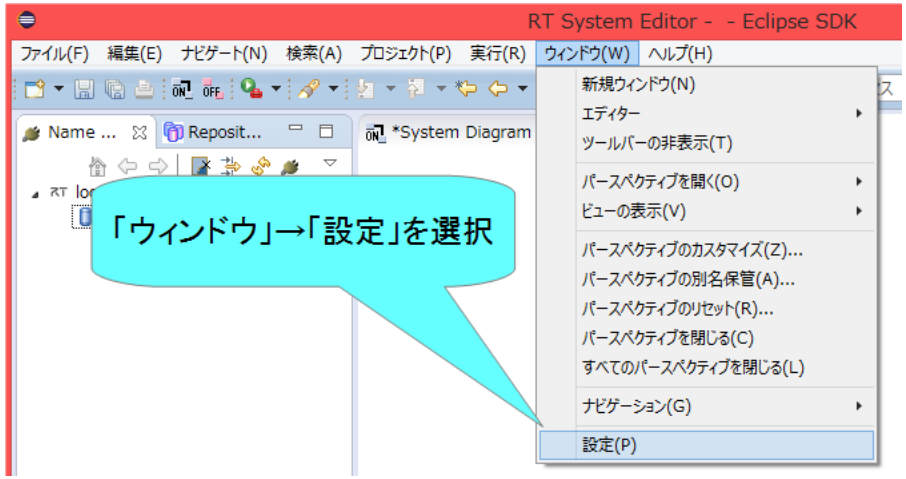
Set up service ports

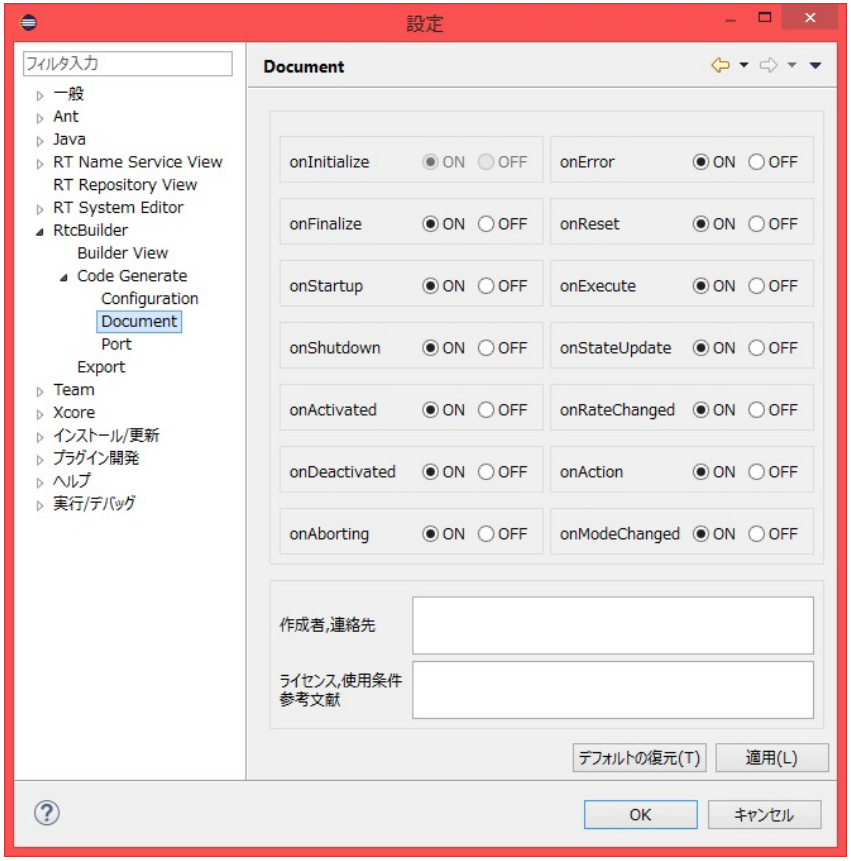
- About IDL files
  - Interface definition languages independent of programming languages
  -

```
1  module SimpleService {
2    typedef sequence<string> EchoList;
3    typedef sequence<float> ValueList;
4    interface MyService
5    {
6          string echo(in string msg);
7          EchoList get_echo_history();
8          void set_value(in float value);
9          float get_value();
10         ValueList get_value_history();
11   };
12 };
```

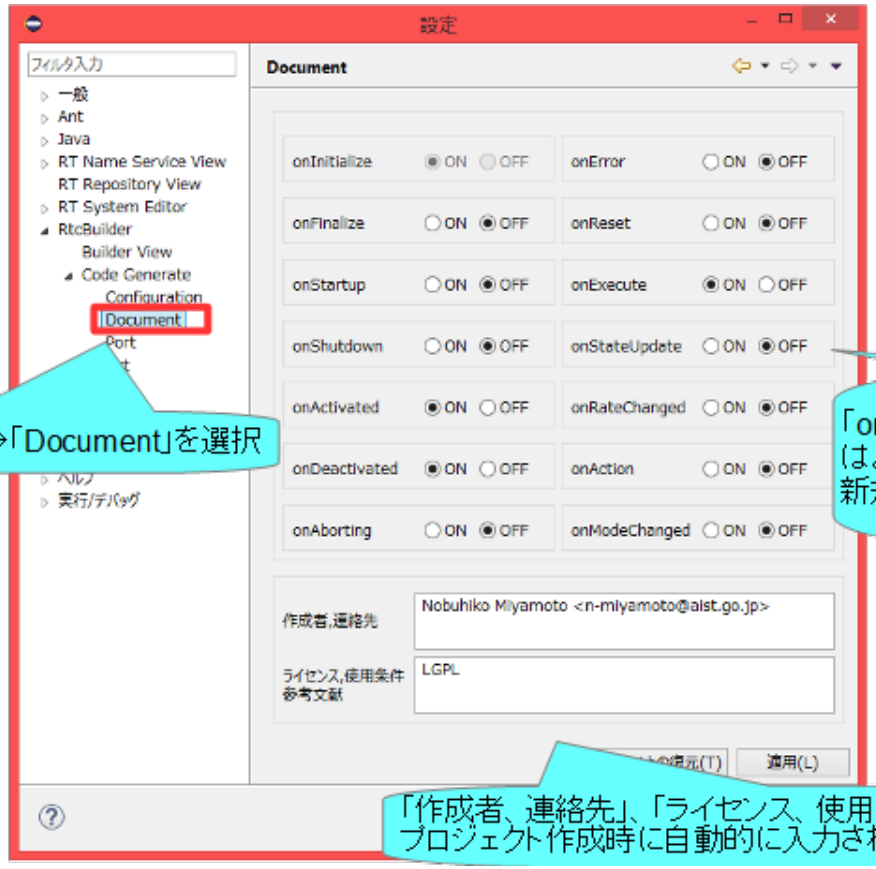
- Call operations such as echo, get_value, etc. on the conschmer side

RtcBuilder Settings





RtcBuilder Settings



「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。

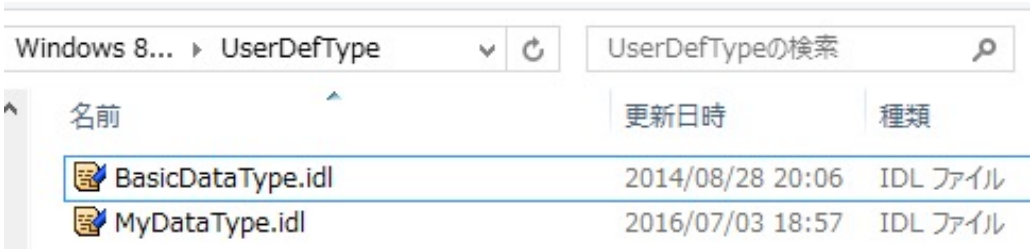
「作成者、連絡先」、「ライセンス、使用条件、参考文献」を入力しておくとプロジェクト作成時に自動的に入力されるので便利。

Use your own data types

- Steps to communicate data ports with your own data types
  - Create an IDL file
    - Create MyDataType.idl in a folder (Here C:¥UserDefType is it)

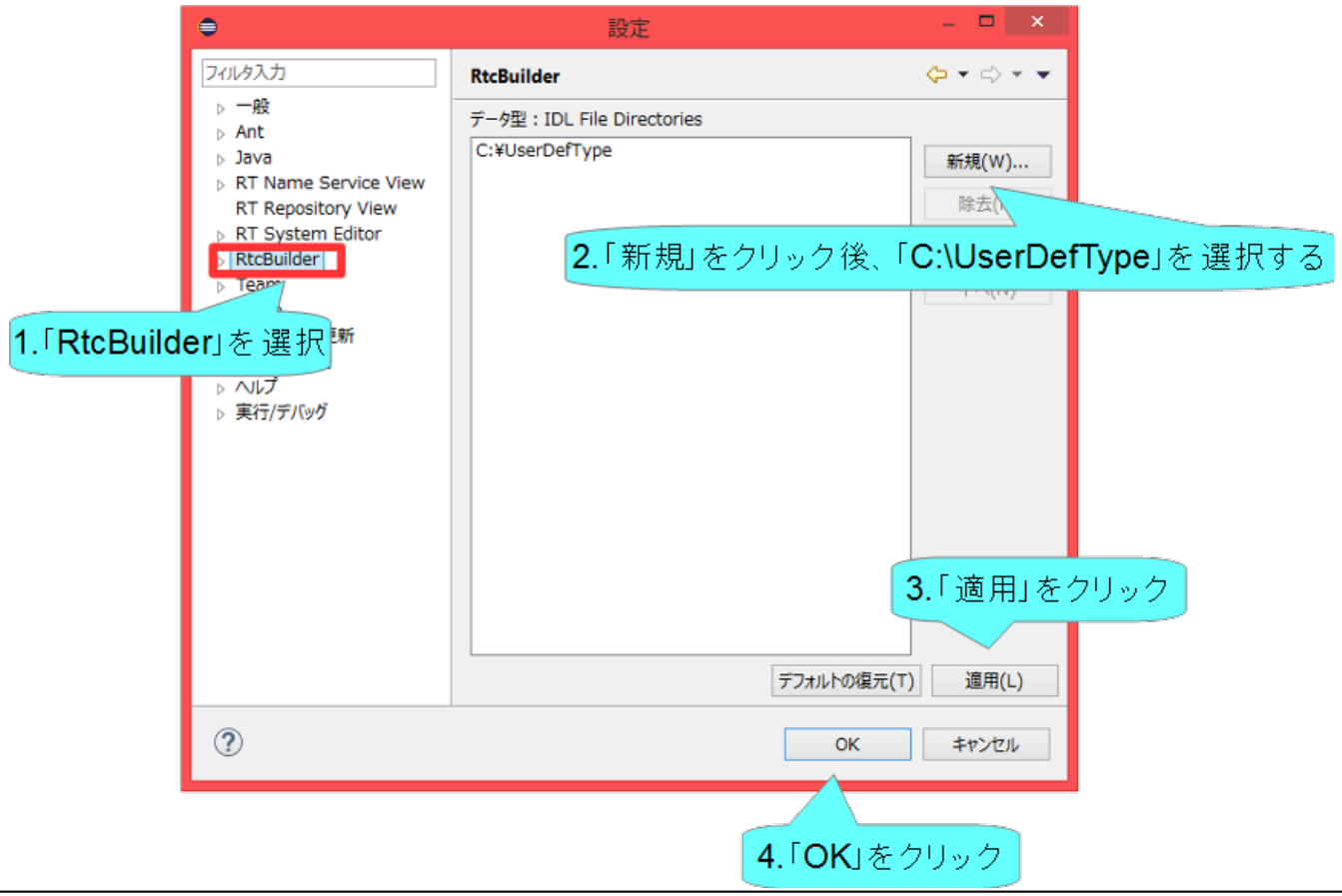
```
1  // @file MyDataType.idl↓
2  #include "BasicDataType.idl"↓
3   ↓
4  struct MyData↓
5  {↓
6    RTC::Time tm;↓
7    short shortVariable;↓
8    long longVariable;↓
9    sequence<double> data;↓
10 };[EOF]
```

    - Copy to the same folder if you are includeing another IDL file



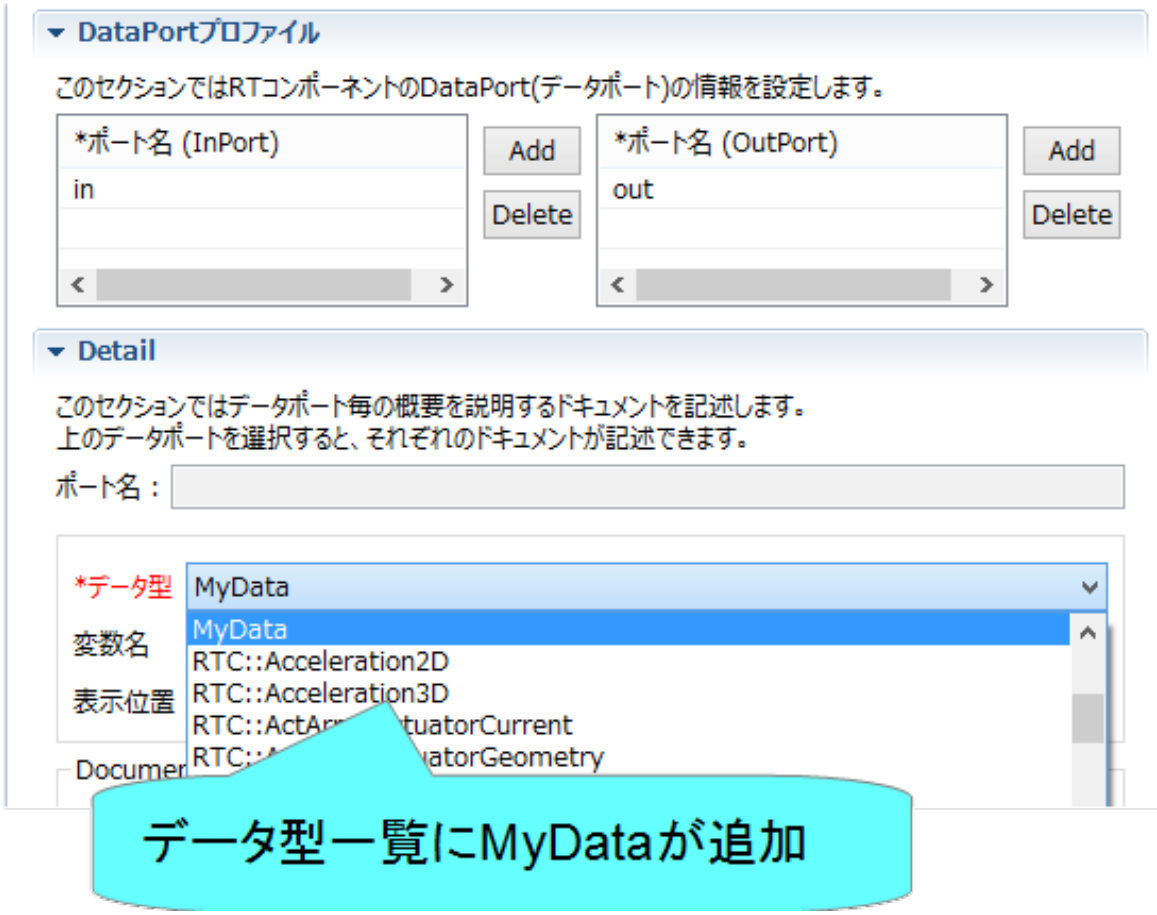
Use your own data types

- Steps to communicate data ports with your own data types
  - Add directory with IDL files in RTC Builder settings



Use your own data types

- Steps to communicate data ports with your own data types



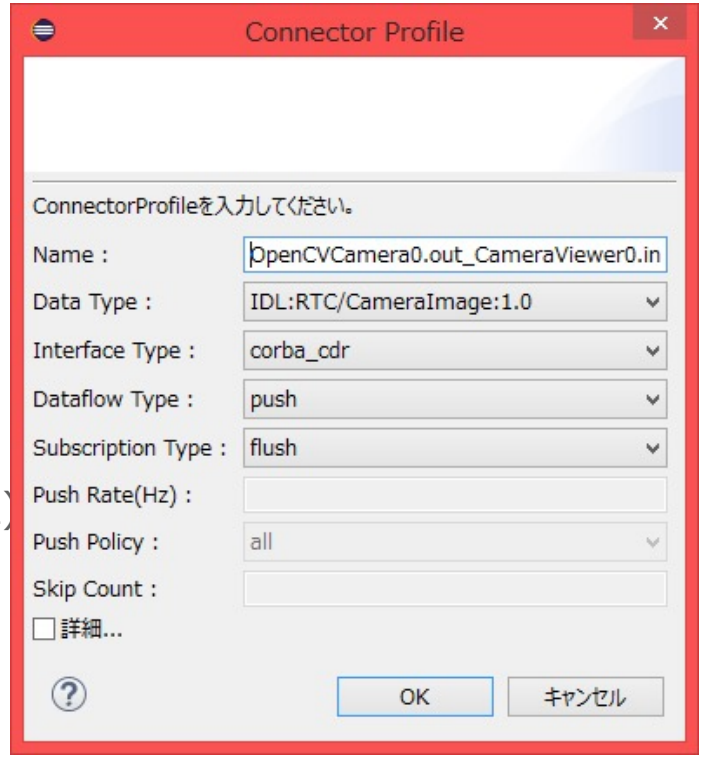
RTSystemEditor supplement explanation

Set up connector profiles

Item	Settings
Name	Name of the connection
DataType	Data type communication between ports ex)TimedOctet,TimedShort, etc.
InterfaceType	Data send method. ex)corba_cdr, etc.
DataFlowType	Data flow type. ex)push, pull, etc.
SubscriptionType	Data transmission timing. It is valid only DataFlowType is push. Options are New, Periodic, Flush.
Push Rate	Freq. of data transimission(Hz). It is only valid if th eSubscriptionType is Periodic.
Push Policy	Data send policy. It is valid if SubscriptionType is New or Periodic. Options are all, fifo, skip, new.
Skip Count	Skip count of data send cycle. It is valid If Push Policy is Skip.

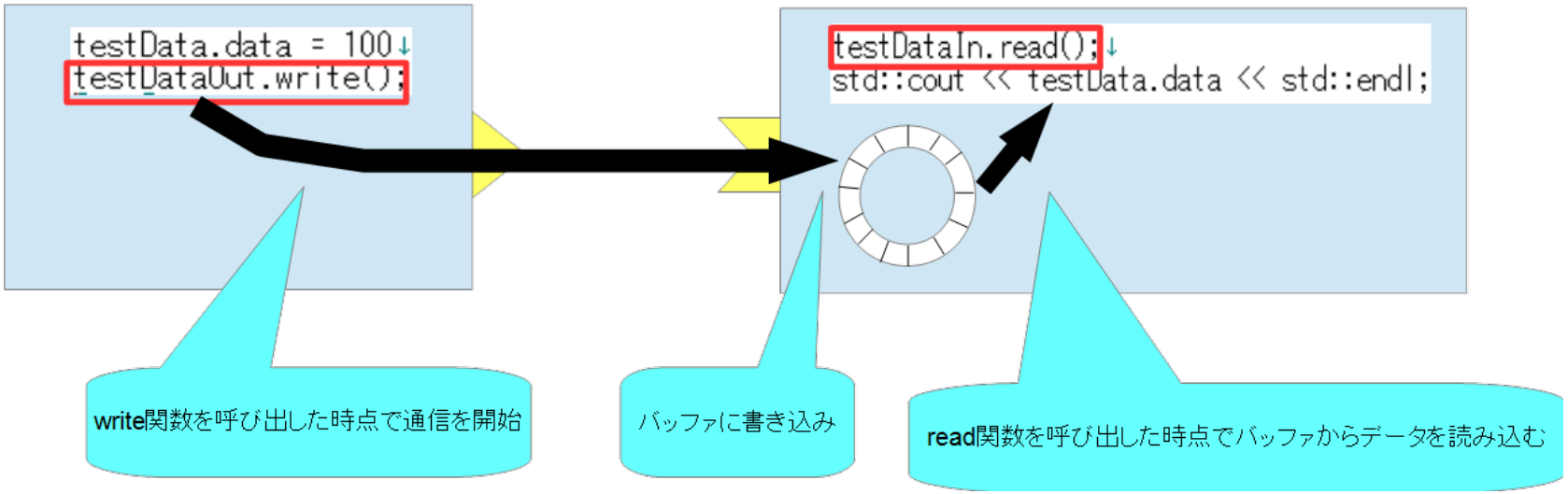
Set up connector profiles

- InterfaceTye
  - Data send scheme type
  - 1.2.0: The following types are added.
    - Direct (valid if RTCs are in the same process)
    - shared_memory (shared memory com)
- DataFlowType
  - Process of data transimission
    - Push
      - Data is push by OutPort to InPort
    - Pull
      - InPort retrieves data from OutPort
- SubscriptionType
  - Data transmission timing (valid if DataFlowType is Push型)
    - flush(Synchronize)
      - Synchronized transmission without buffering
    - new(asynchronous)
      - Send data from a buffer if new data arrived.
    - periodic(asynchronous)
      - —Data transmitted in periodically
- Push Policy(SubscriptionType if new or periodicのみ)
  - Data transimission policy
    - all
      - Bulk send data in a buffer
    - fifo
      - Send data in the buffer one by one with FIFO
    - skip
      - Send between data in a buffer
    - 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)

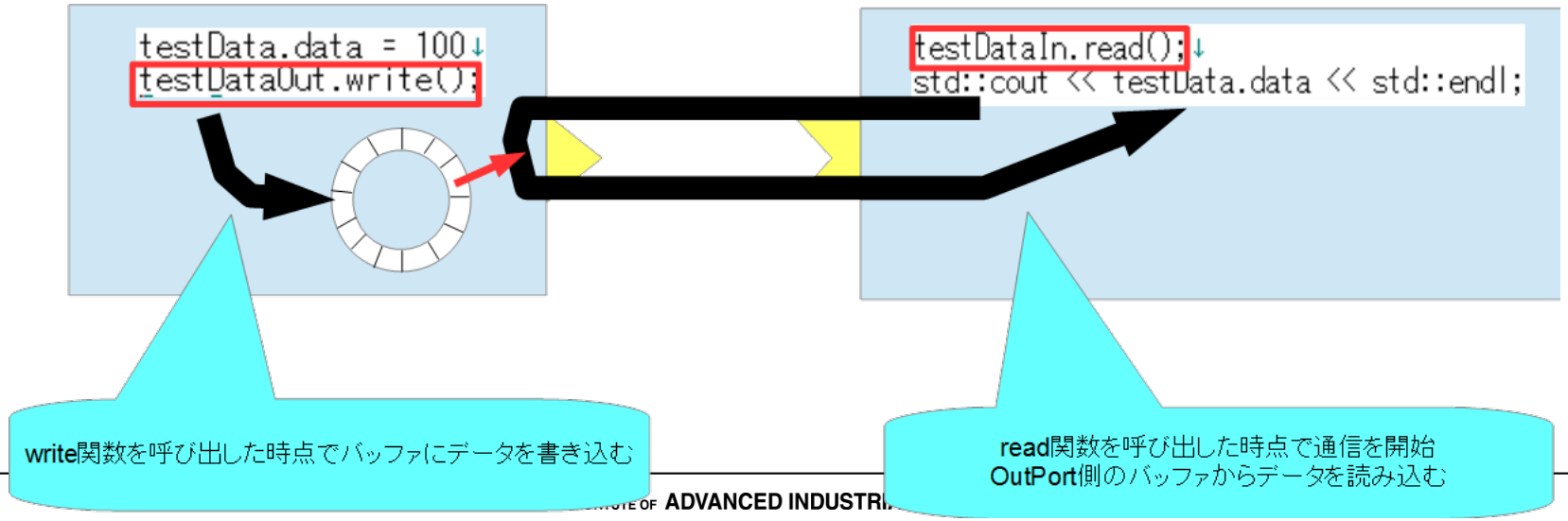


Set up connector profiles

- DataFlowType
  - Push

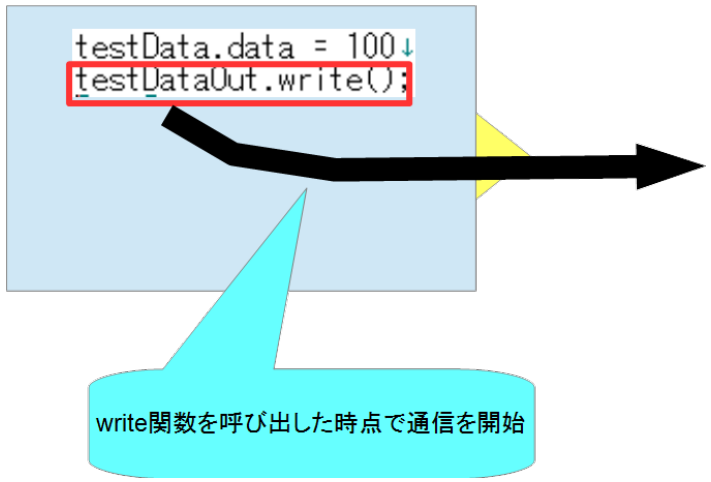


  - Pull

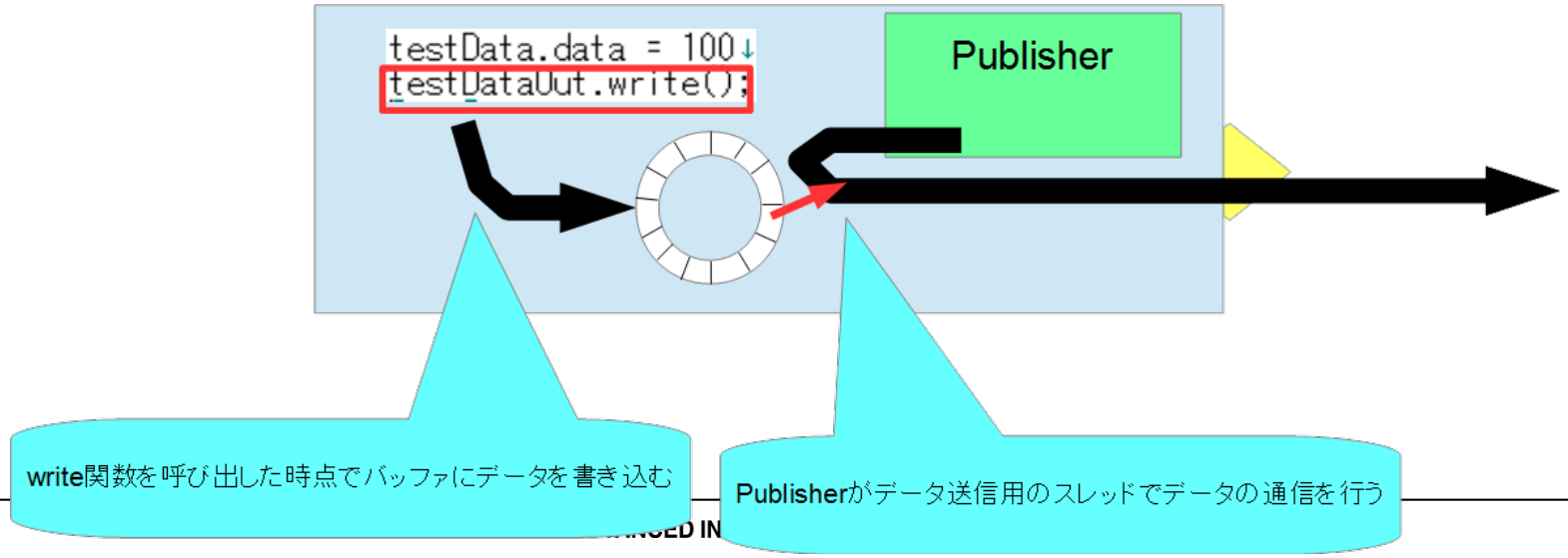


Set up connector profiles

- SubscriptionType
  - flush(同期)

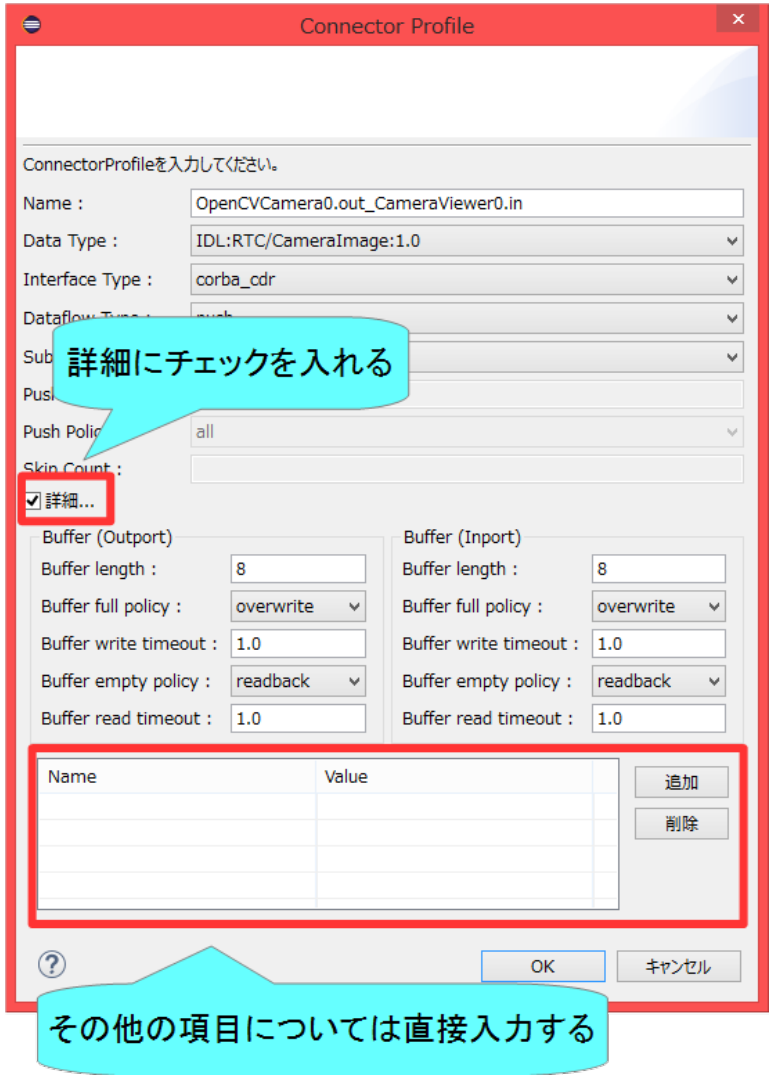


  - new、 periodic(非同期)



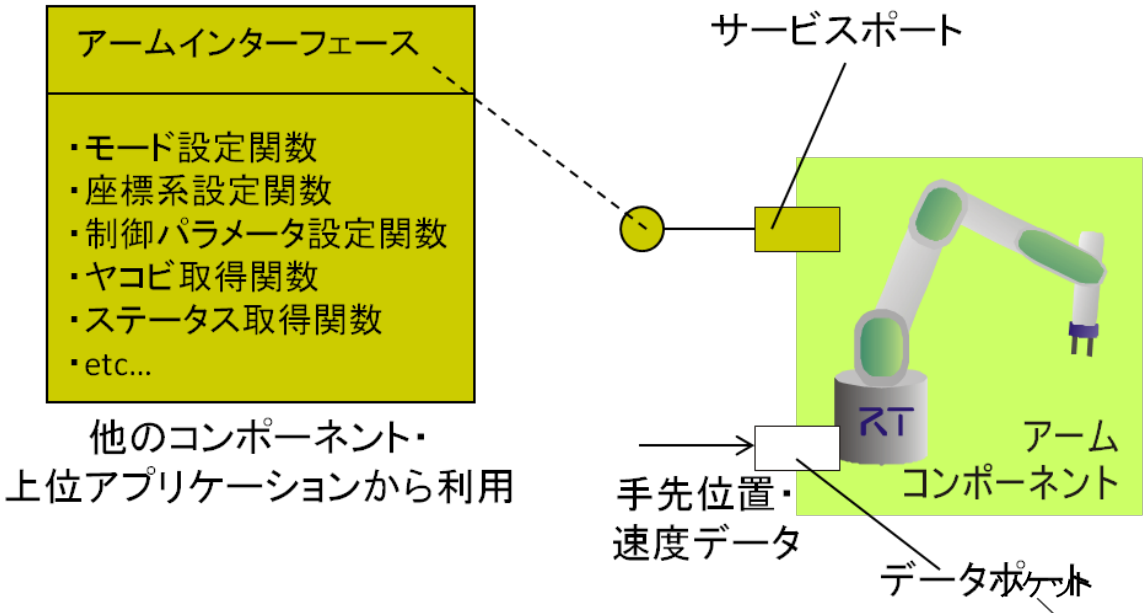
Set up connector profiles

Item	Setting
Buffer length	Length of the buffer
Buffer full policy	What to do if it is bufferful when writing data. (overwrite, do_nothing, block)
Buffer write timeout	Time of timeout event when writing data (s)
Buffer empty policy	What to do if the buffer is empty when reading data. (readback, do_nothing, block)
Buffer read timeout	Time of timeout event when reading data (s)

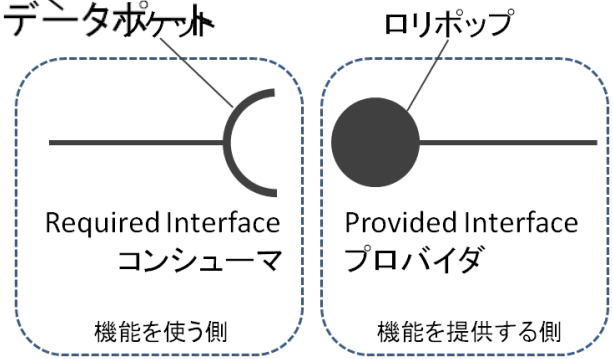


サービスポートについて

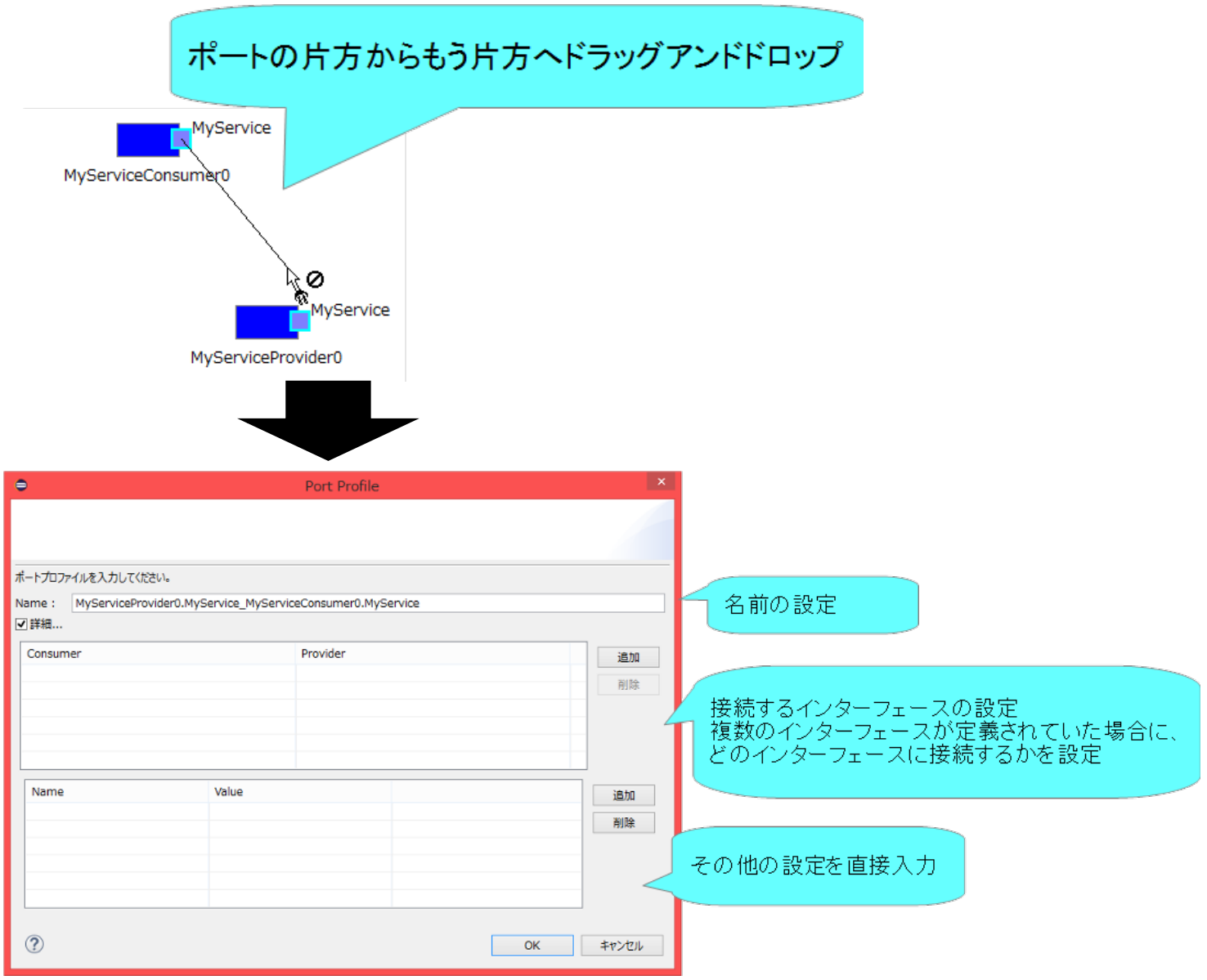
- コマンドレベルのやり取りを行うための仕組み
  - 任意のタイミングで操作を行いたい時などに使用
    - 例えばロボットアームのサーボを停止させる、ハンドを閉じる等



- コンシューマ側がプロバイダ側が提供する関数群(オペレーション、メソッド)を呼び出す
- インターフェースはIDLファイルで定義する。

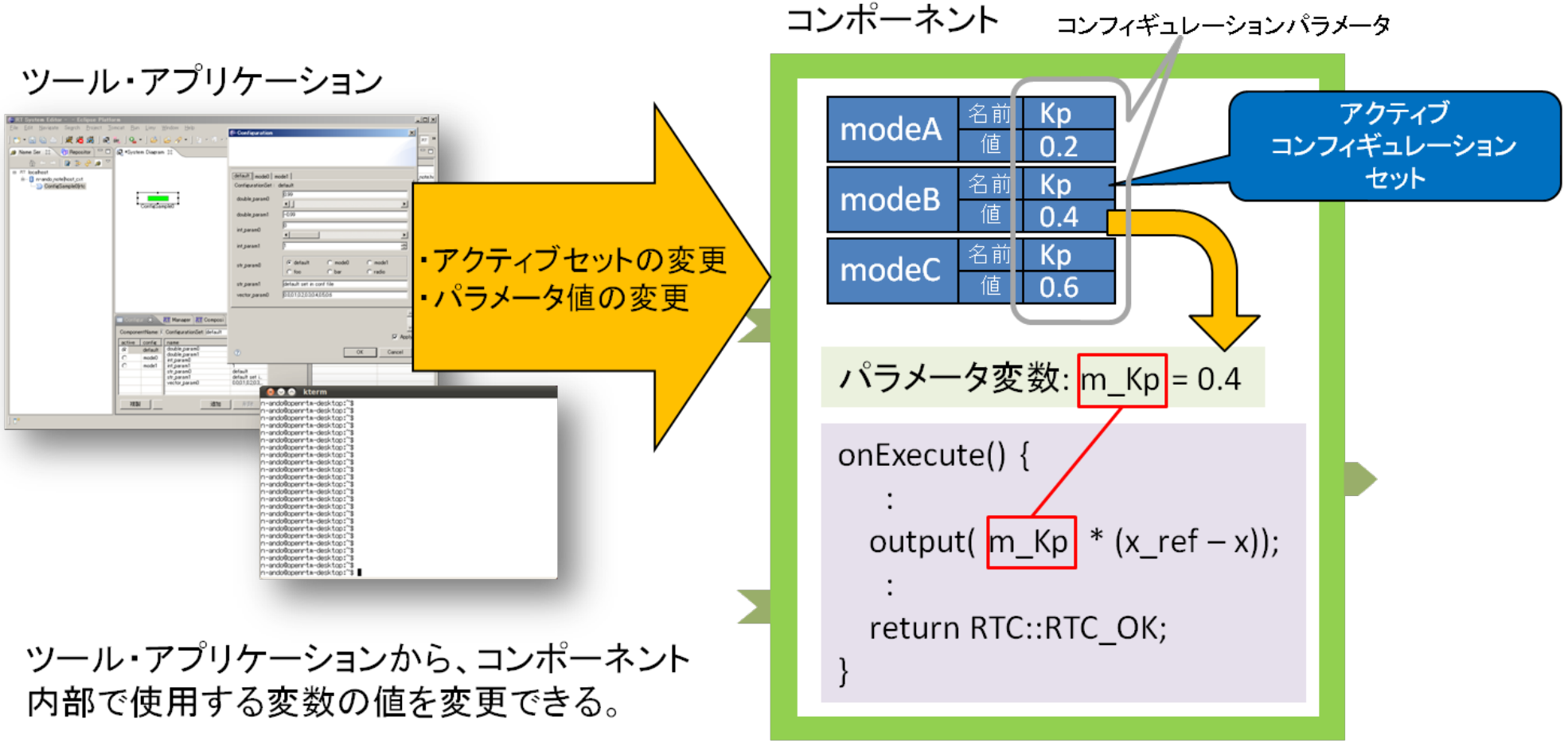


サービスポートの接続

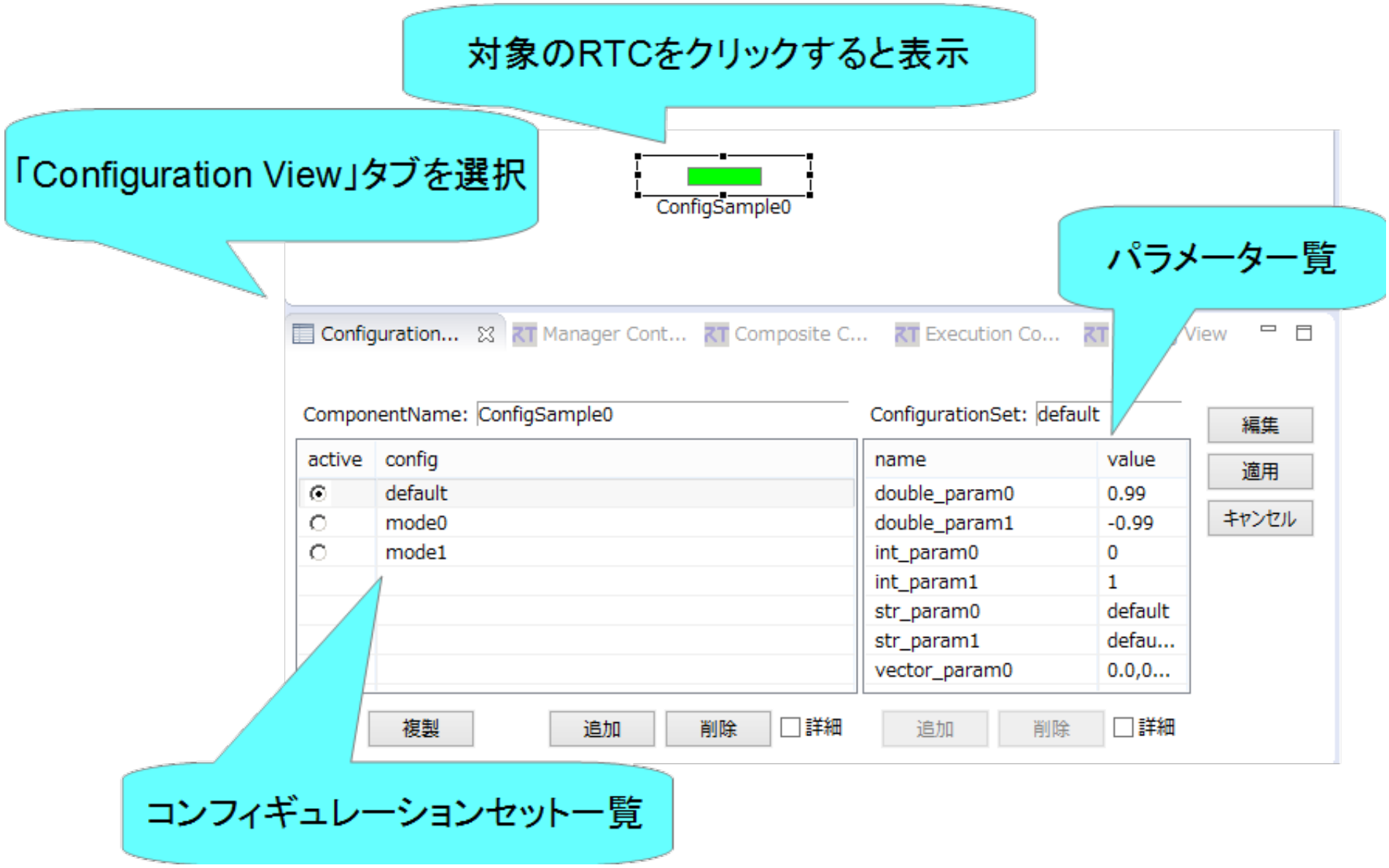

ポートの片方からもう片方へドラッグアンドドロップ
MyService
MyServiceConsumer0
MyService
MyServiceProvider0
Port Profile
ポートプロファイルを入力してください。
Name :
MyServiceProvider0.MyService_MyServiceConsumer0.MyService
詳細...
Consumer
Provider
追加
削除
Name
Value
追加
削除
OK
キャンセル
名前の設定
接続するインターフェースの設定
複数のインターフェースが定義されていた場合に、
どのインターフェースに接続するかを設定
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について

- パラメータを外部から操作する仕組み
  - コンポーネント作成後に変更が必要なパラメータを設定する
    - 例えばデバイスが接続されているCOMポート番号の設定等

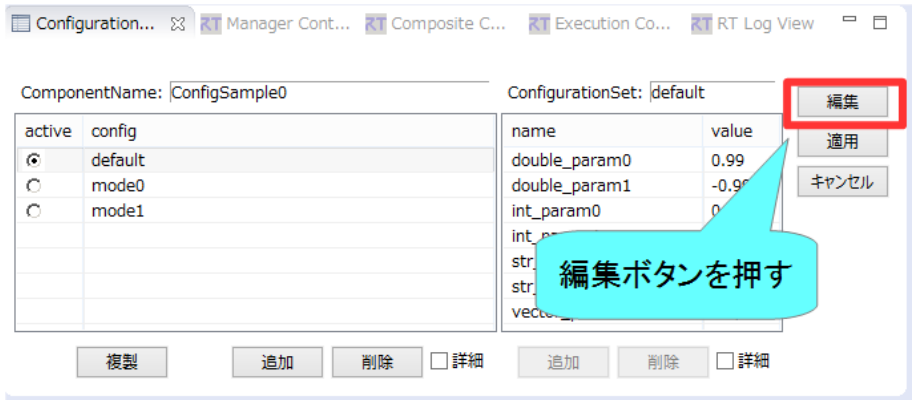


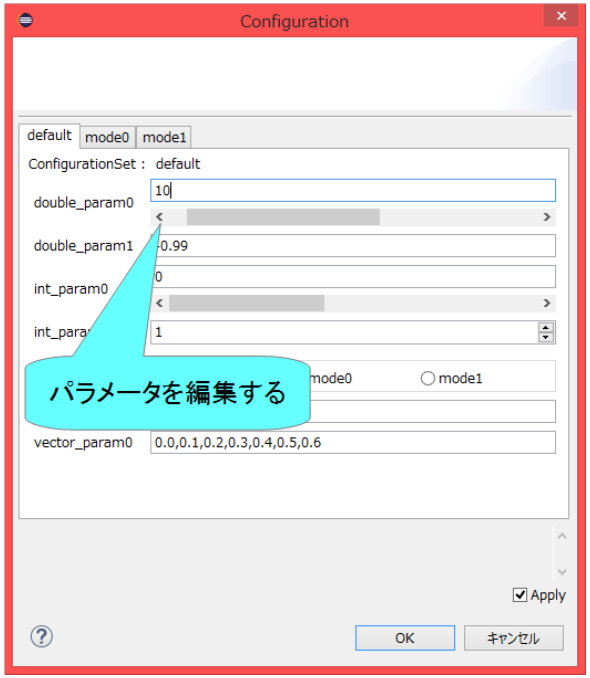
コンフィギュレーションパラメータの設定



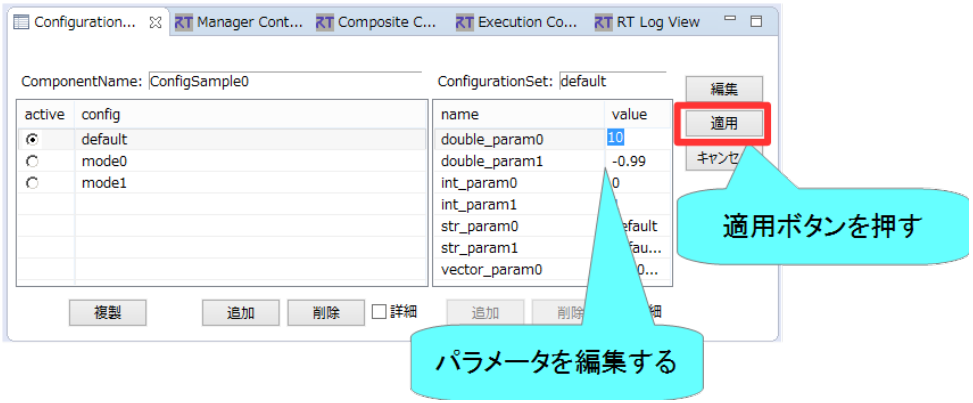
コンフィギュレーションパラメータの設定

- 方法1



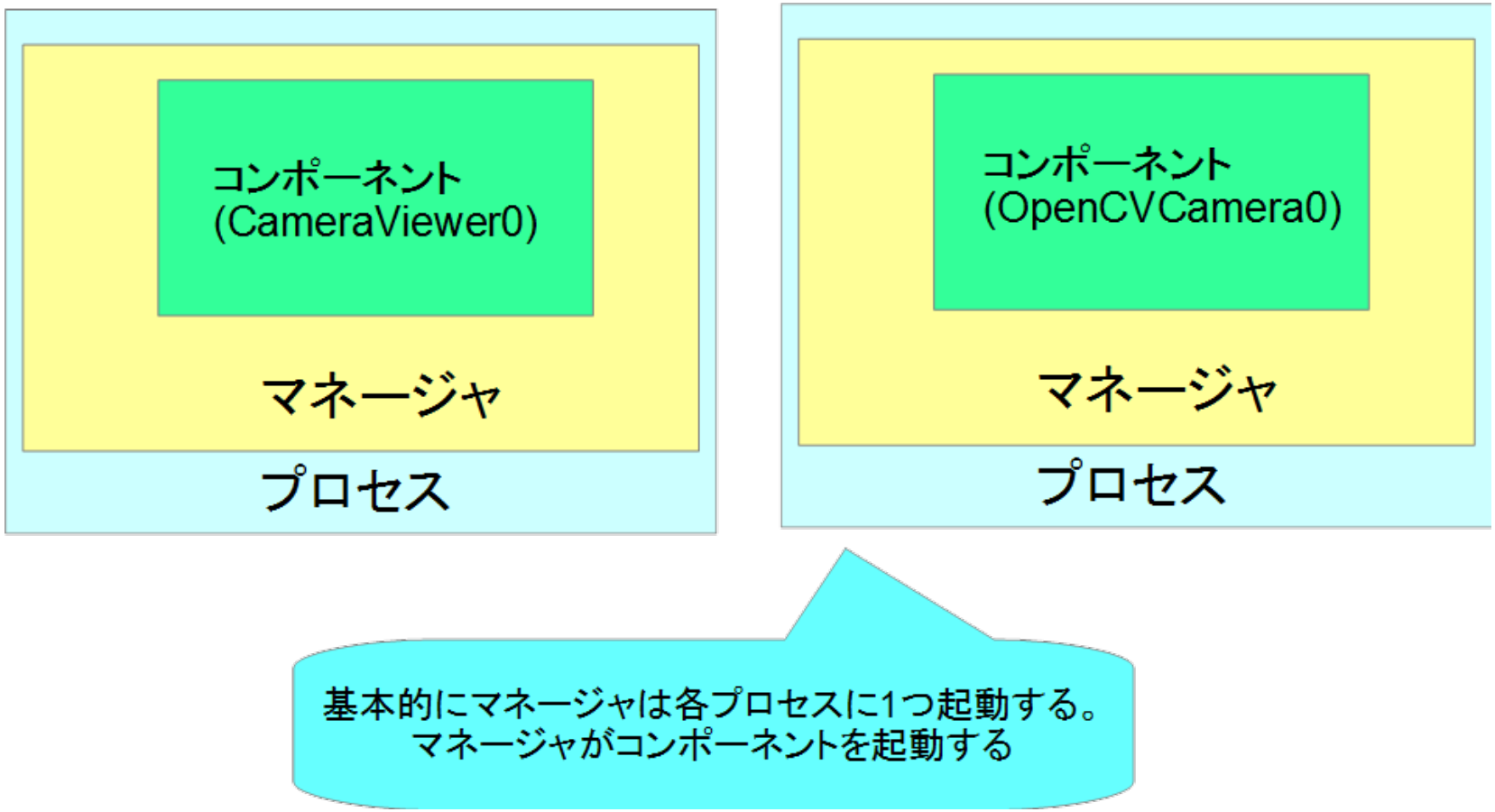


- 方法2

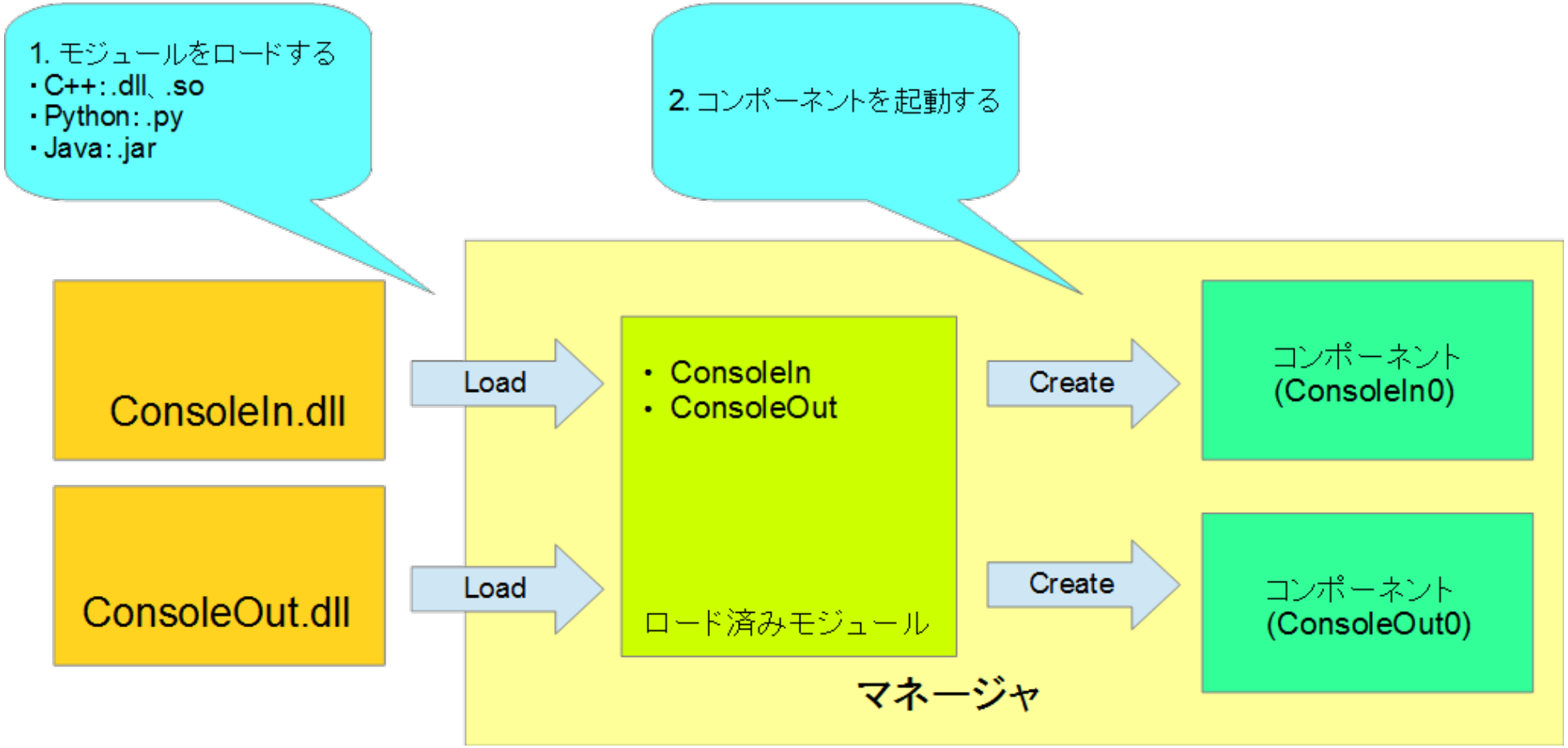


マネージャの操作

- CameraViewerComp.exe、OpenCVCameraComp.exeのプロセスではマネージャが起動している
  - マネージャがコンポーネントを起動する

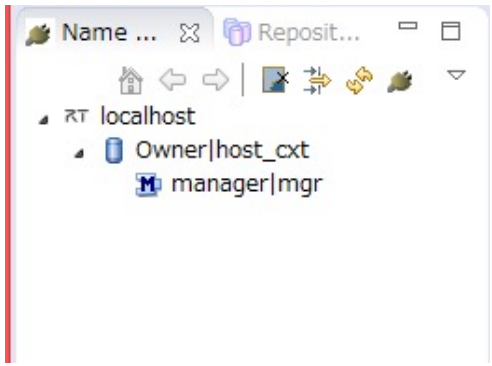


マネージャの操作


1. モジュールをロードする
・C++: .dll、.so
・Python: .py
・Java: .jar
2. コンポーネントを起動する
ConsoleIn.dll
ConsoleOut.dll
Load
Load
・ConsoleIn
・ConsoleOut
ロード済みモジュール
Create
Create
コンポーネント
(ConsoleIn0)
コンポーネント
(ConsoleOut0)
マネージャ

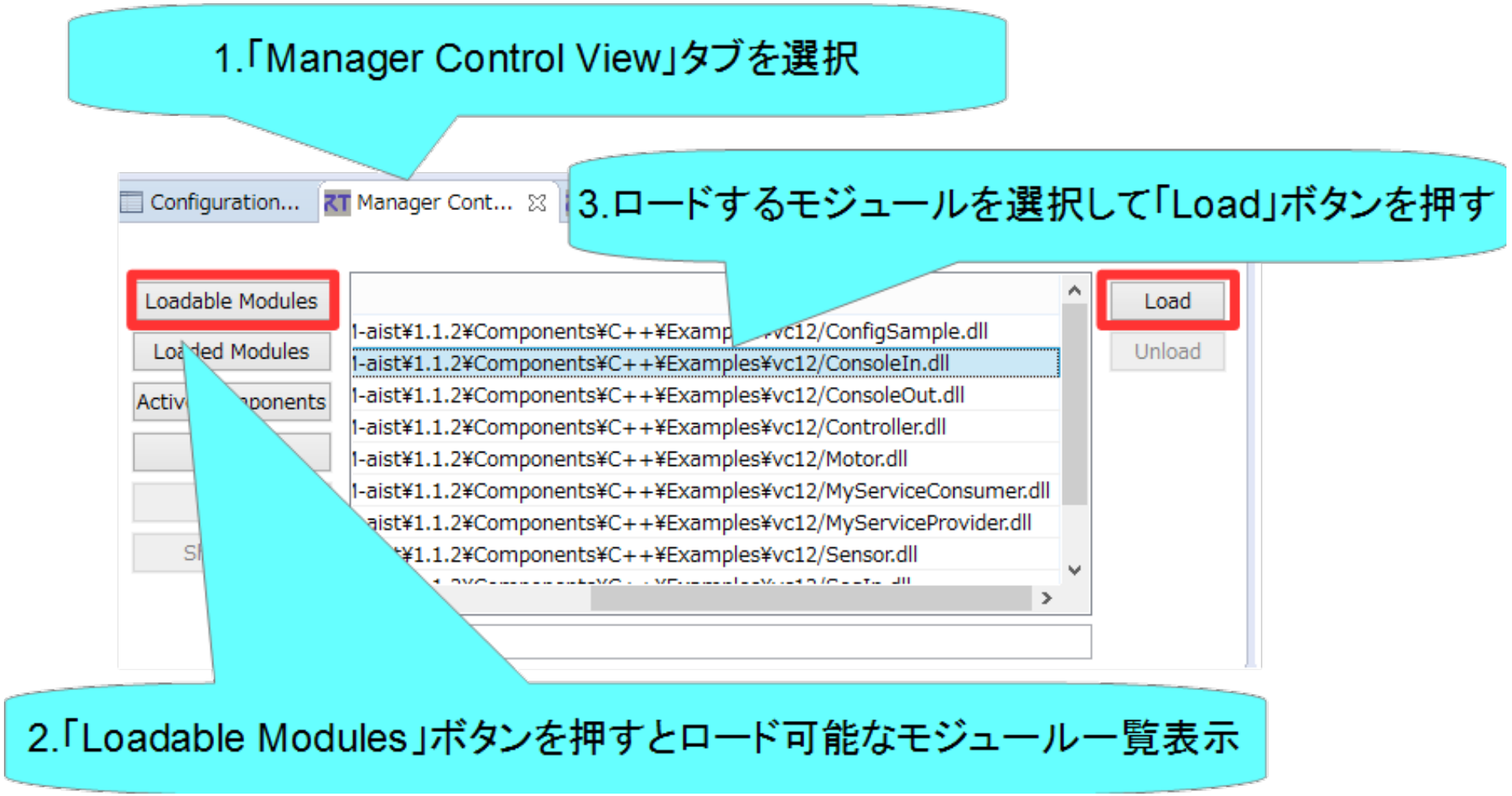
マネージャの操作

- 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
    - rtcdはコマンドプロンプトからrtcd.exeを入力するか、OpenRTM-aistをインストールしたフォルダからコピーして使用する
    - rtcdはマネージャの起動のみを行う
      - ～Comp.exeは起動時に特定のコンポーネントの起動も行う
    - 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



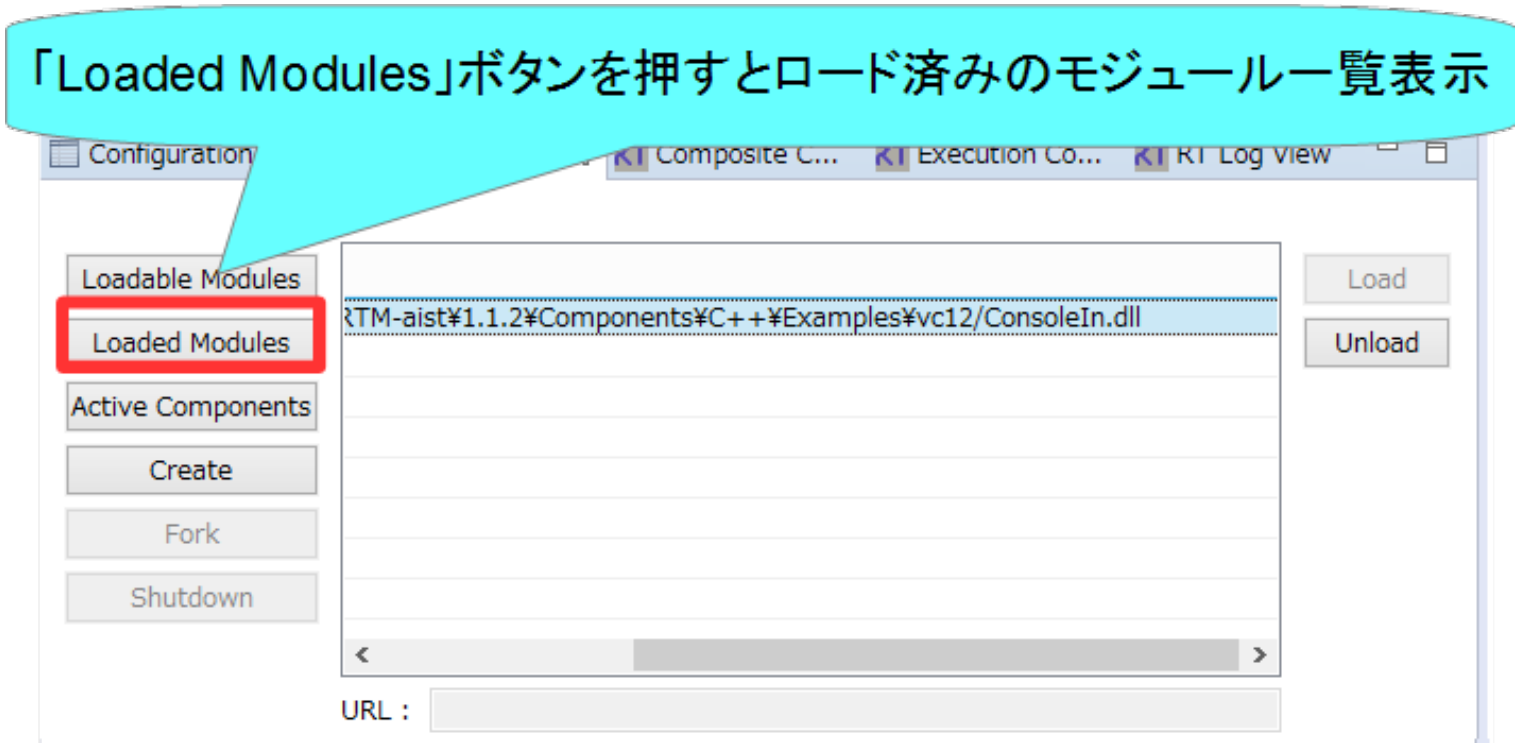
マネージャの操作

- モジュールのロード



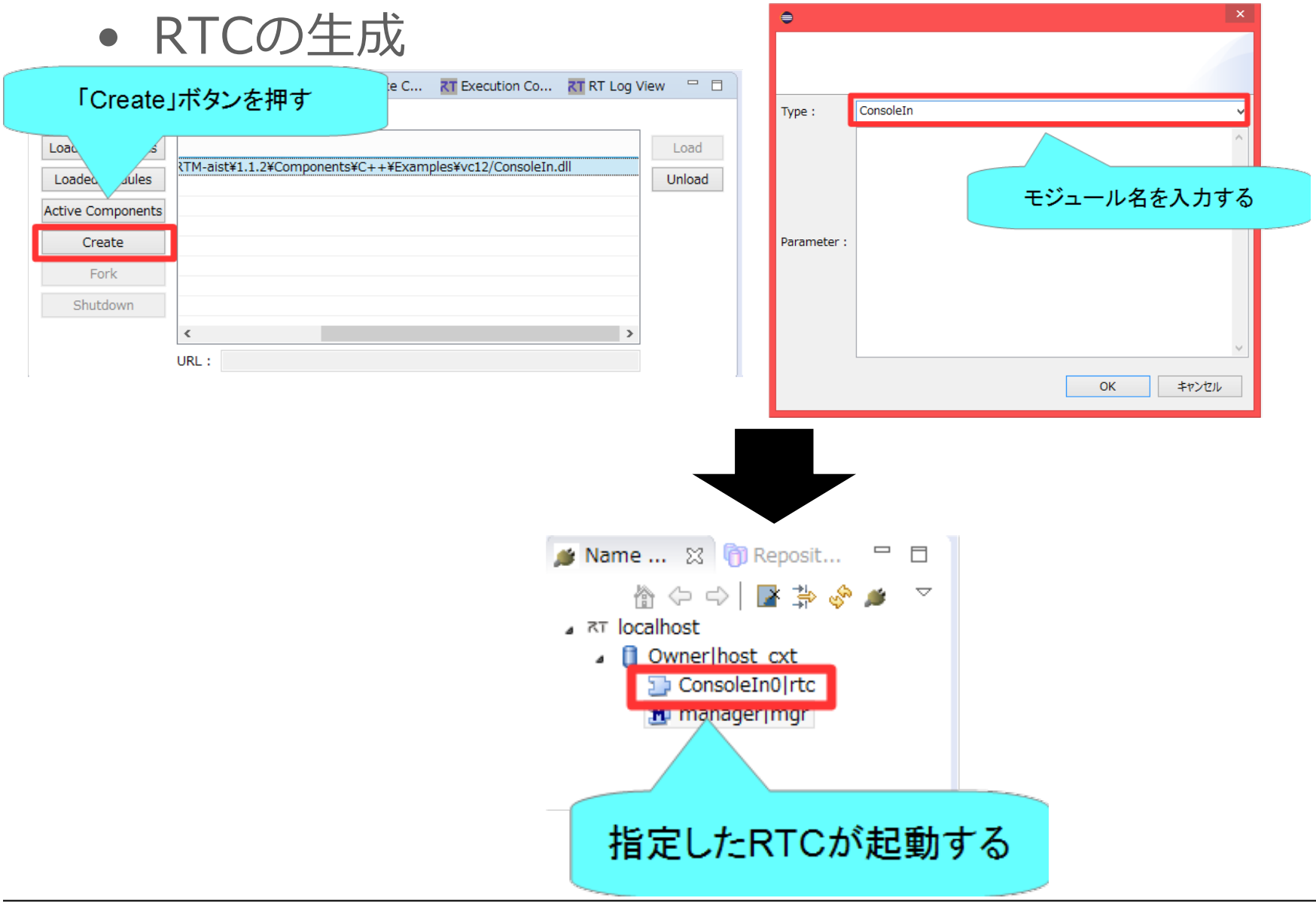
マネージャの操作

- モジュールのロード

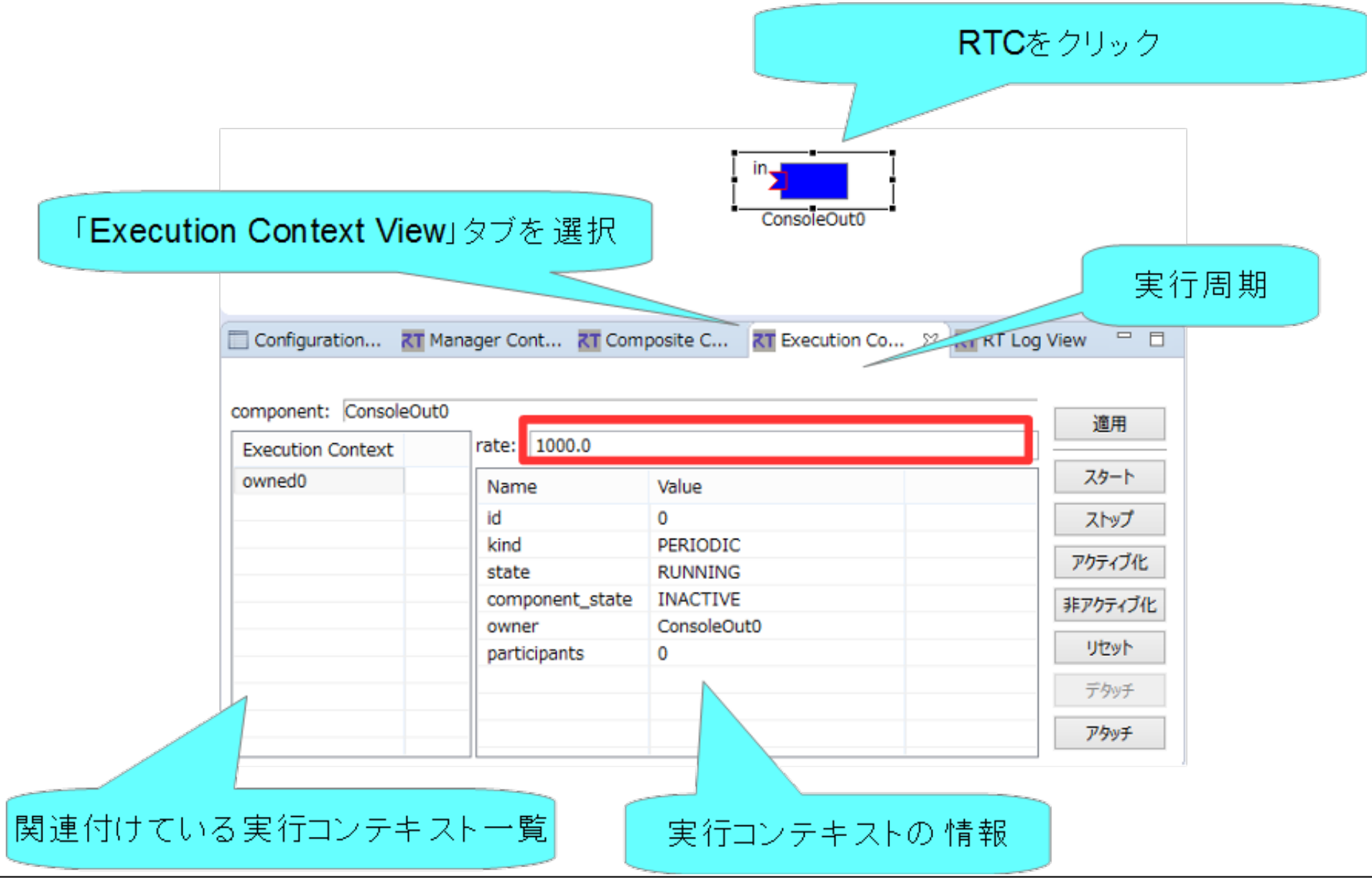


マネージャの操作

- RTCの生成

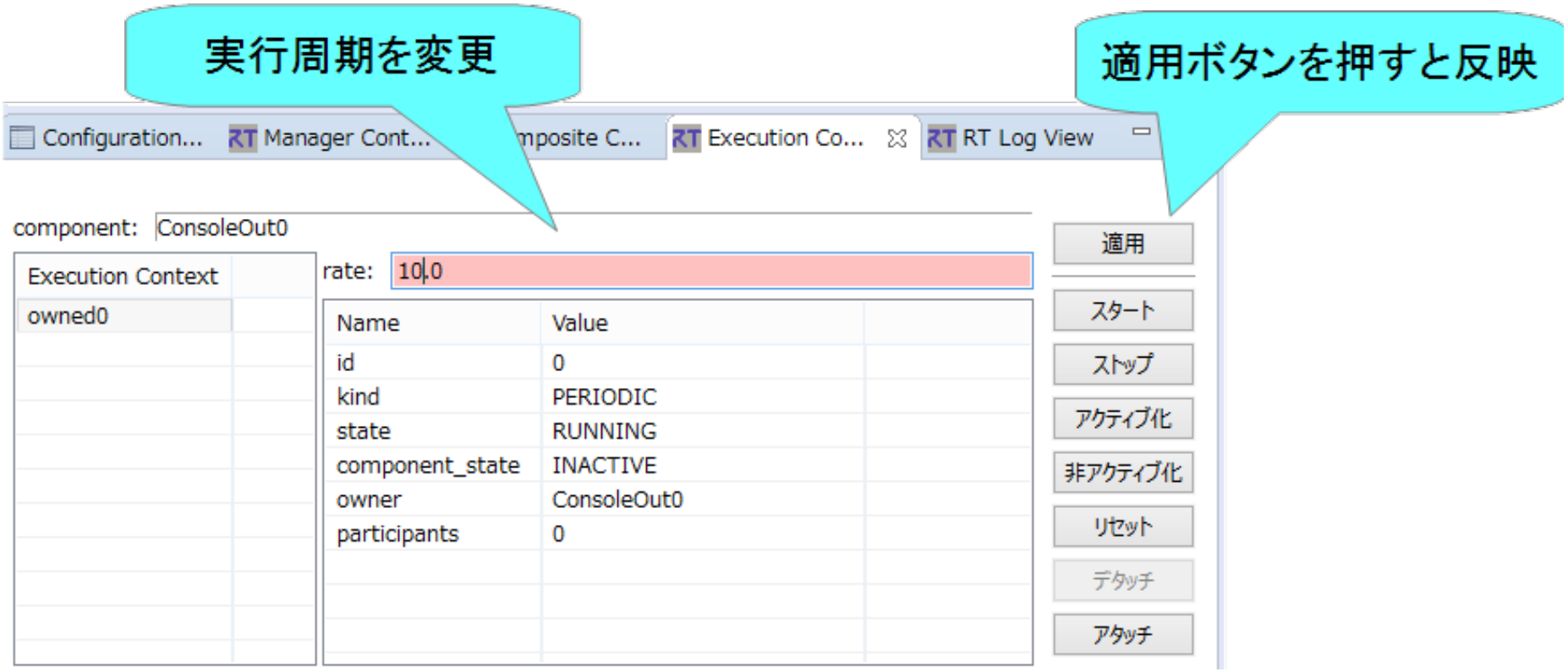


実行コンテキストの操作



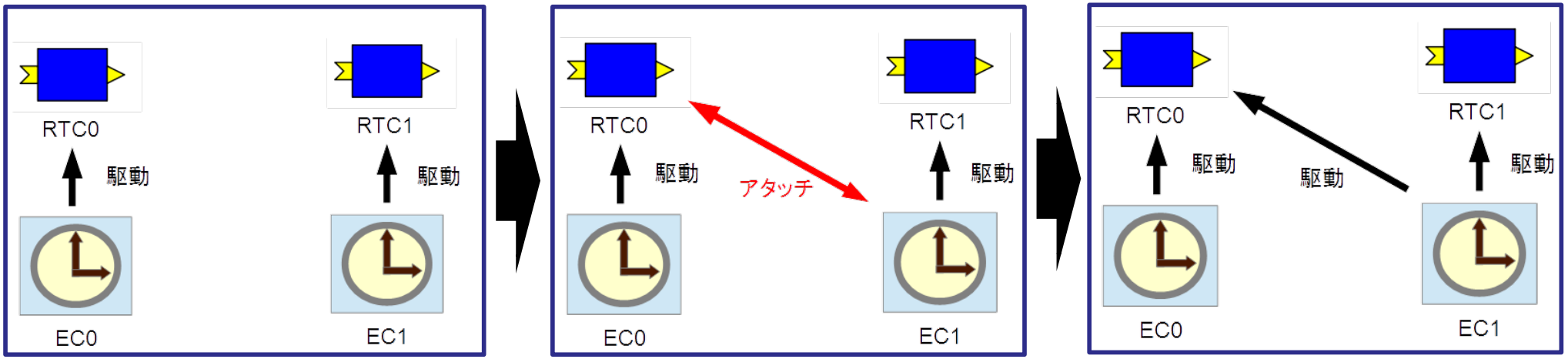
実行コンテキストの操作

- 実行周期の設定



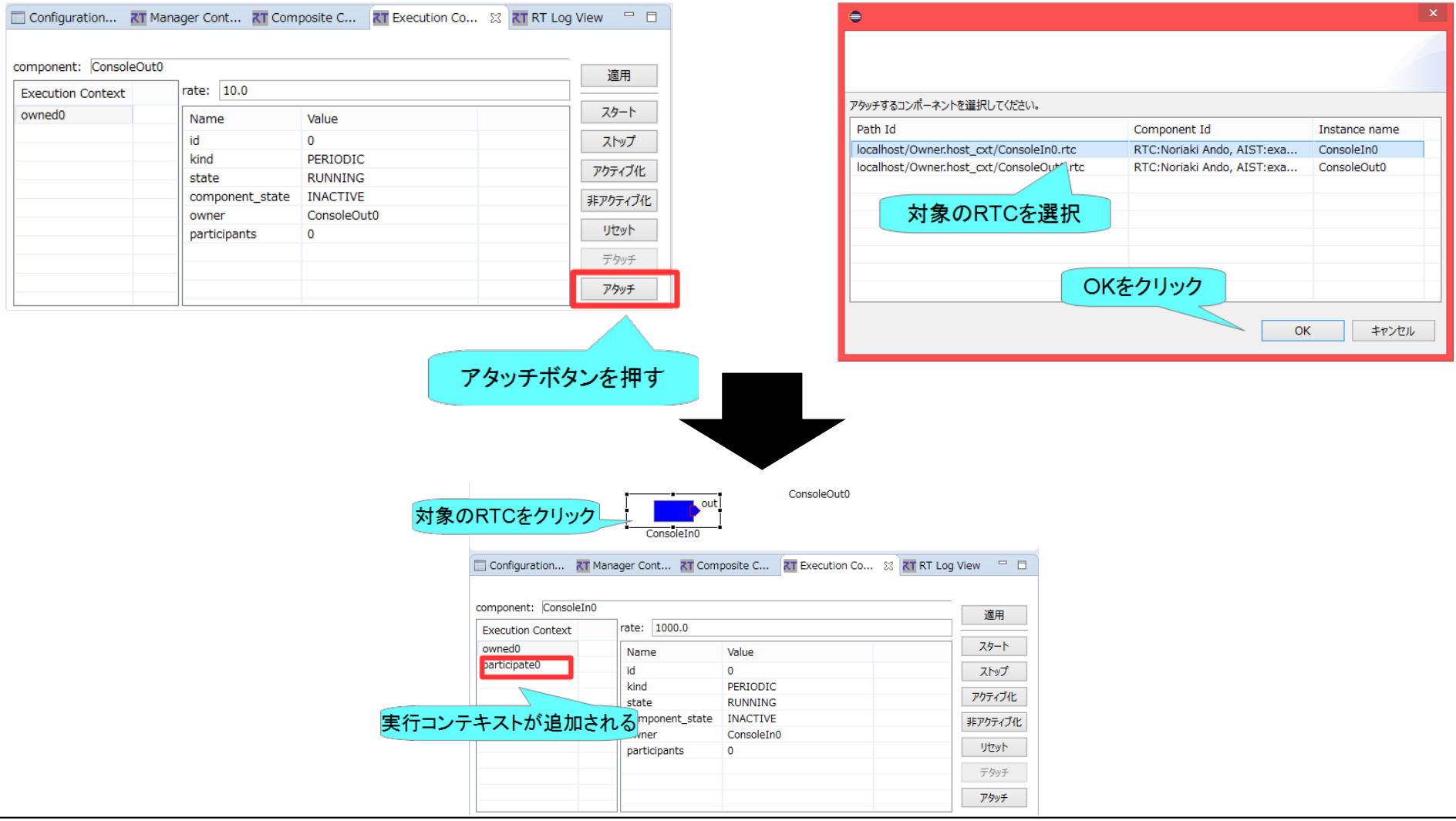
実行コンテキストの操作

- 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



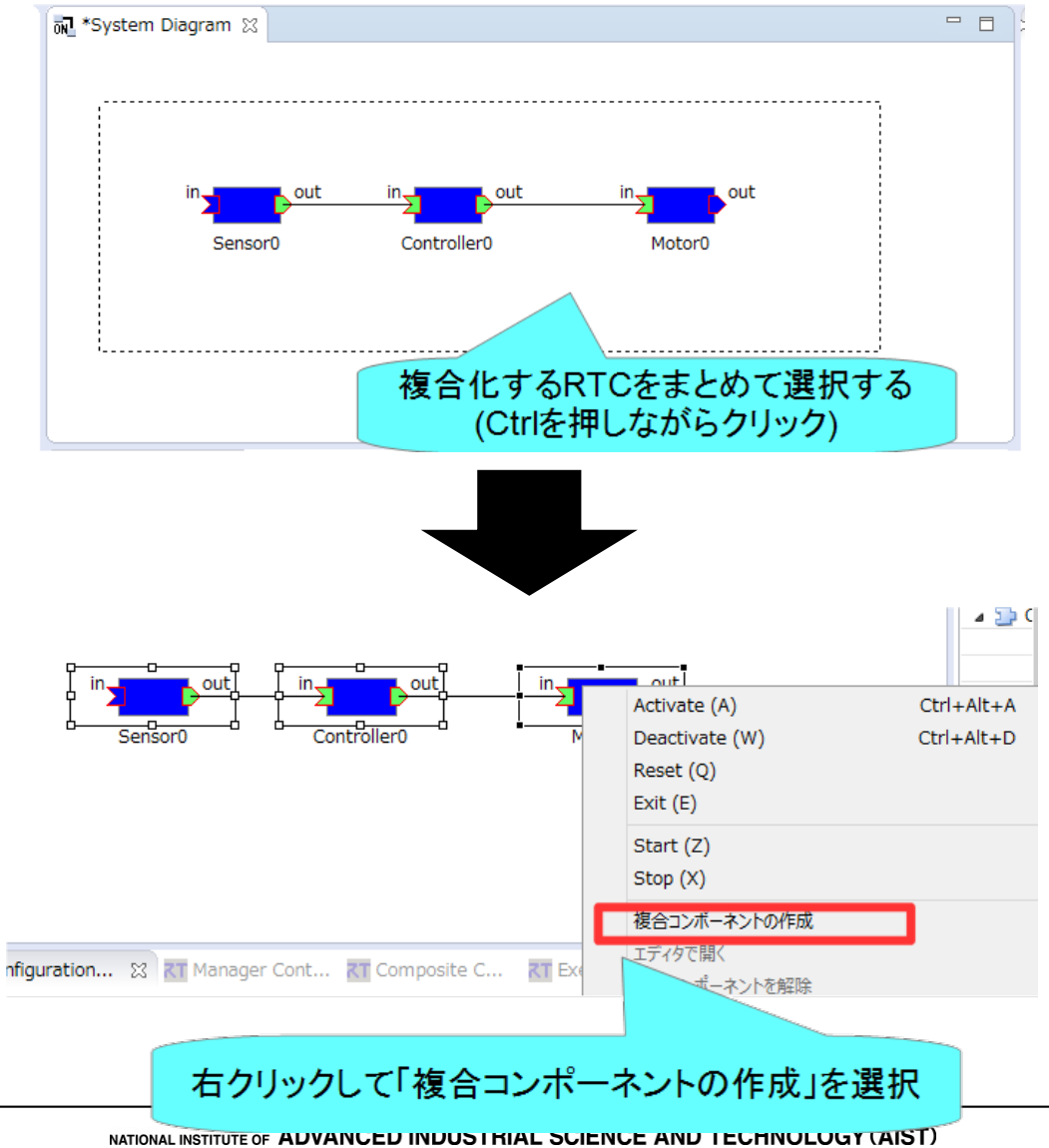
実行コンテキストの操作

- 実行コンテキストの関連付け



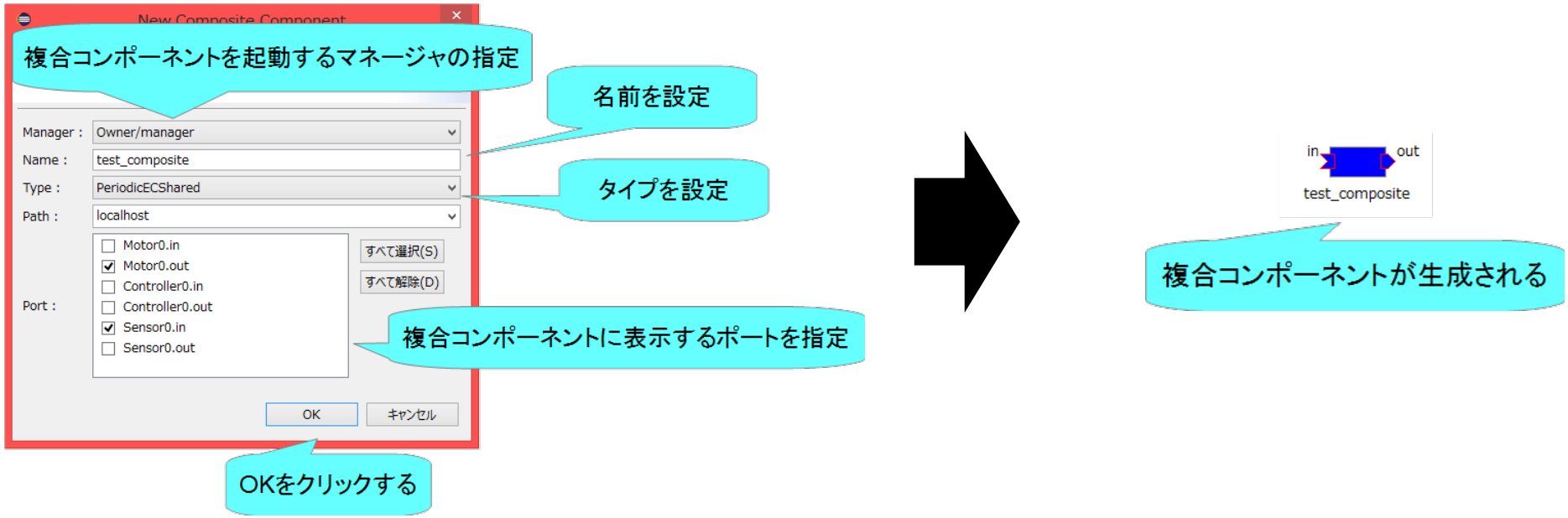
複合コンポーネントの操作

- 複合コンポーネントの生成



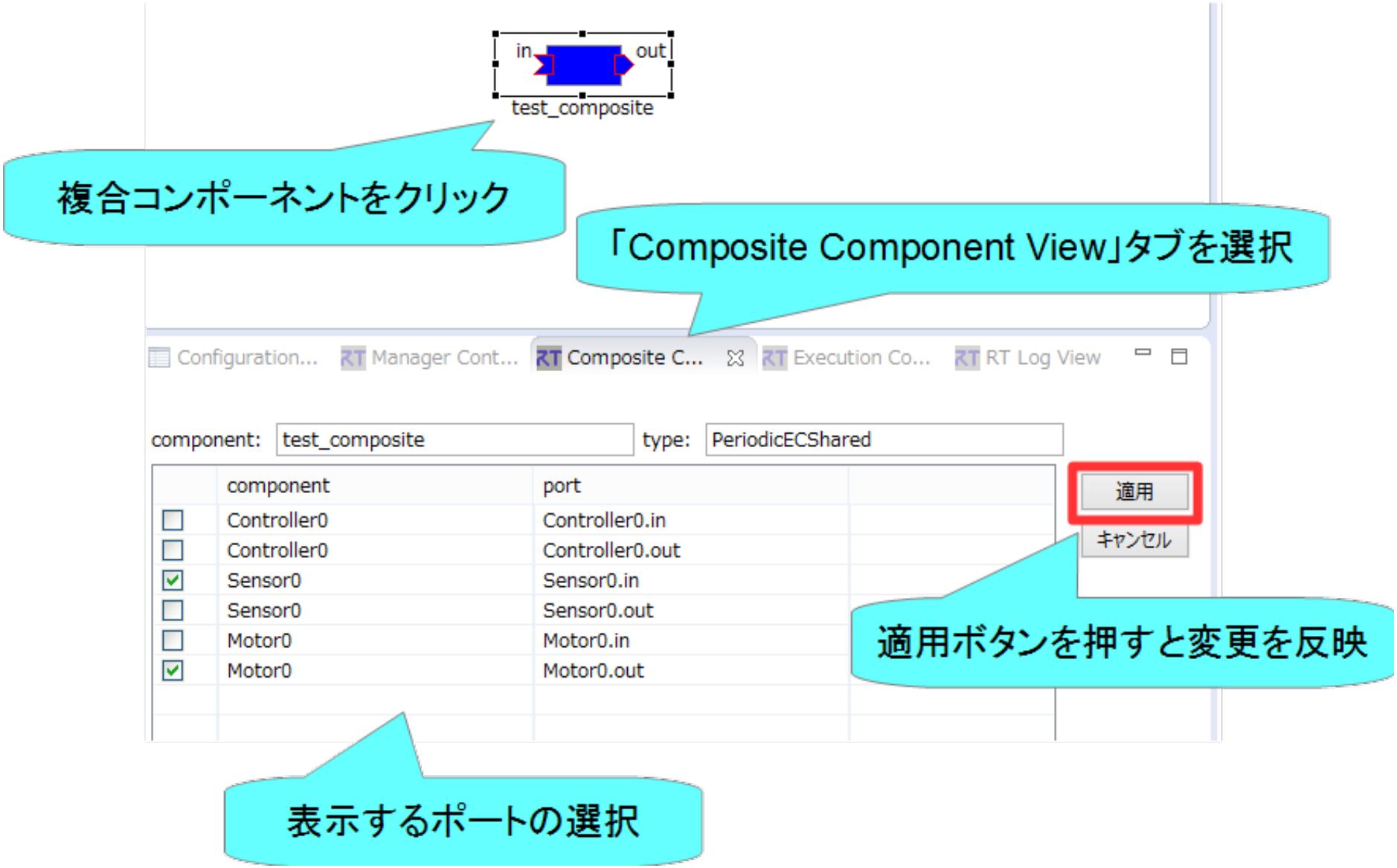
複合コンポーネントの操作

- 複合コンポーネントの生成

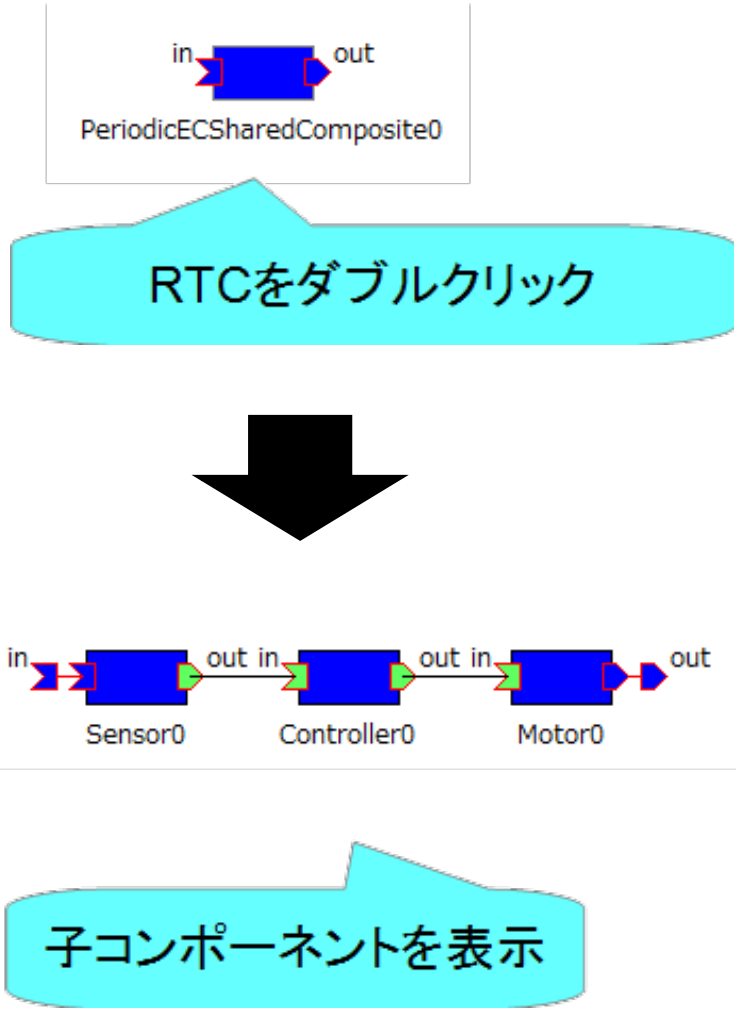


- Type
  - 以下の3種類から選択可能
    - PeriodicECShared
      - 実行コンテキストの共有
    - PeriodicStateShared
      - 実行コンテキスト、状態の共有
    - Grouping
      - グループ化のみ

複合コンポーネントの操作

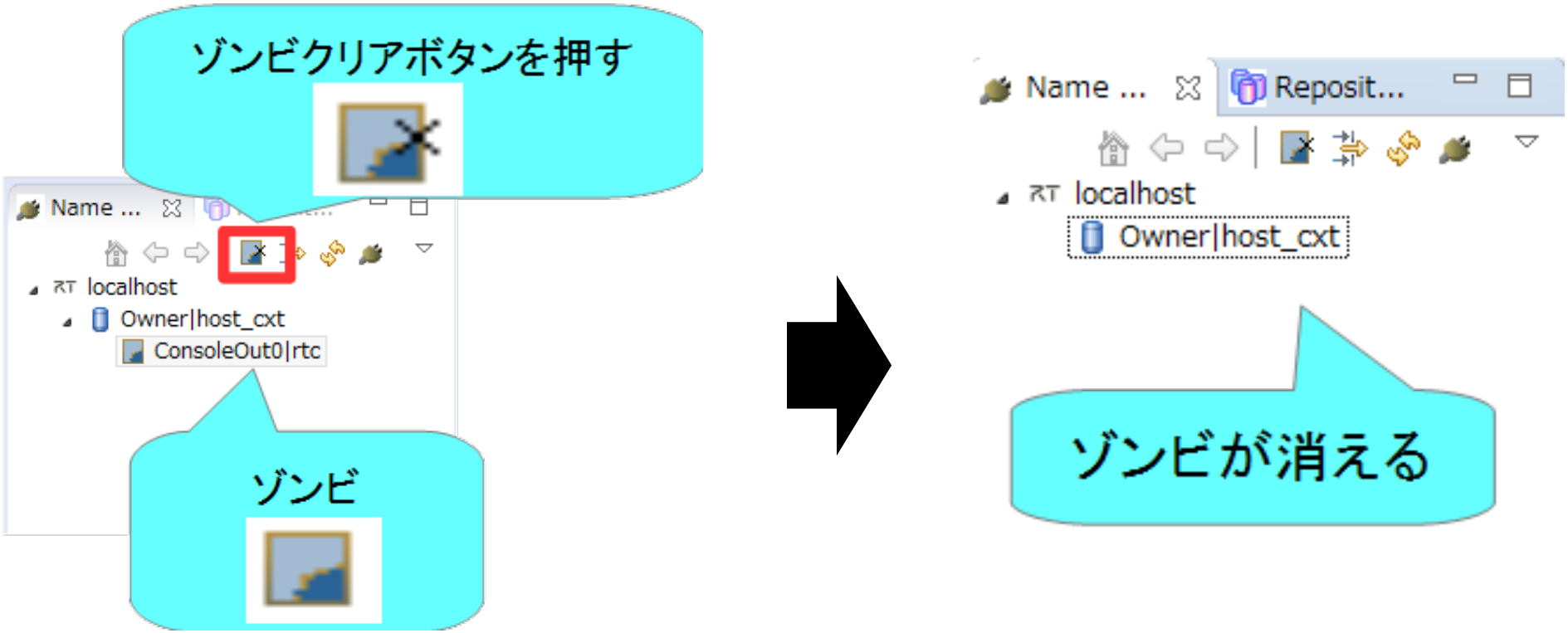


複合コンポーネントの操作



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