

# Part 2

# Getting Started with RT

# Component Creation

Nobuhiko Miyamoto

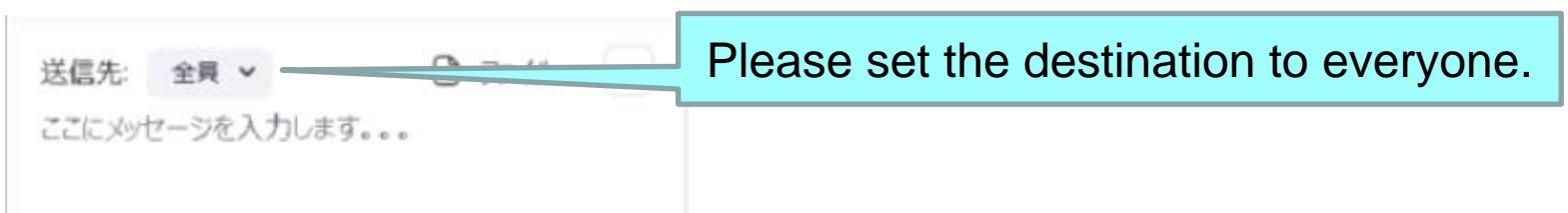
National Institute of Advanced Industrial Science  
and Technology

Industrial Cyber-Physical Systems Research Center  
Software Platform Research Team



# If you have any questions

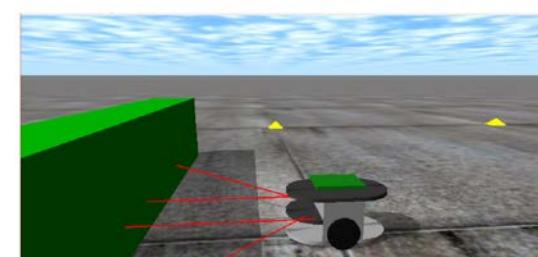
- Ask a question in Zoom's chat.



- If the problem is not solved by chat, we will provide individual support.

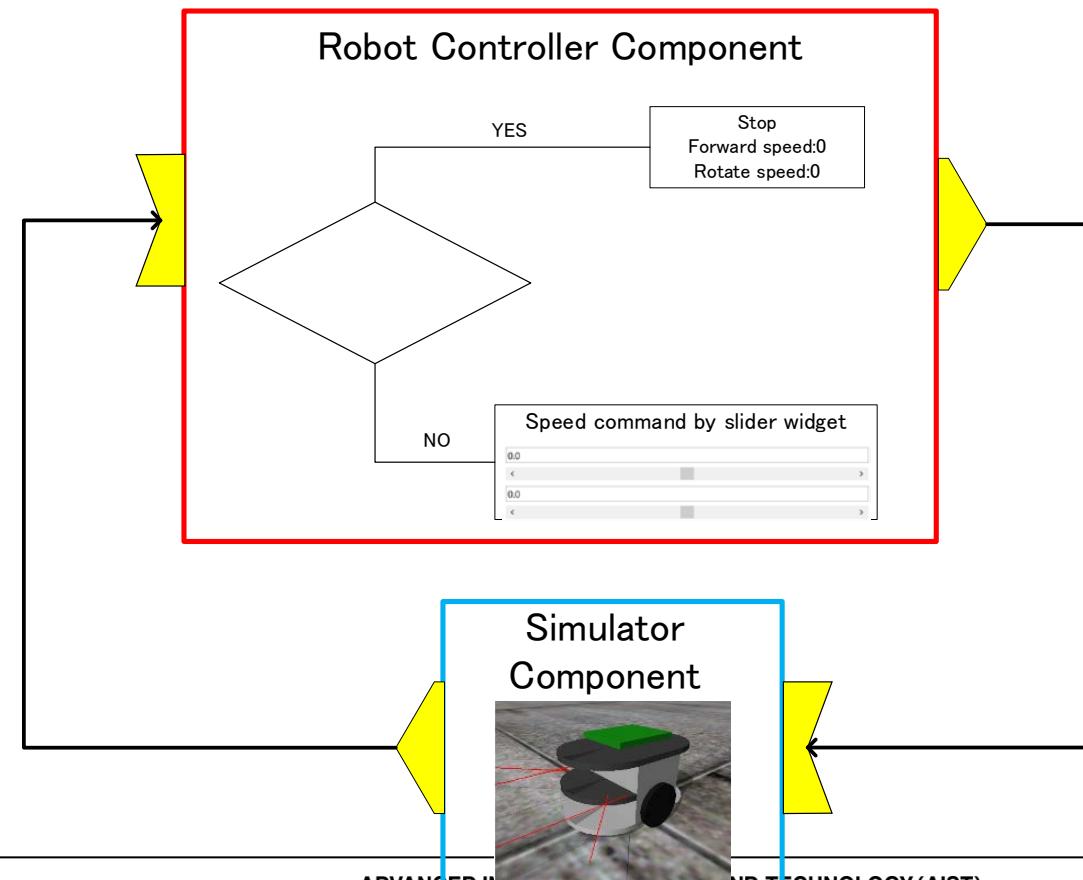
# Document

- Please, open the following WEB page
- Windows
  - [https://openrtm.org/openrtm/ja/doc/casestudy/raspberrypi\\_mouse/raspimouse\\_tutorial\\_rtm\\_seminar/tutorial\\_rtm\\_seminar\\_win\\_part2](https://openrtm.org/openrtm/ja/doc/casestudy/raspberrypi_mouse/raspimouse_tutorial_rtm_seminar/tutorial_rtm_seminar_win_part2)
- Ubuntu
  - [https://openrtm.org/openrtm/ja/doc/casestudy/raspberrypi\\_mouse/raspimouse\\_tutorial\\_rtm\\_seminar/tutorial\\_rtm\\_seminar\\_ubuntu\\_part2](https://openrtm.org/openrtm/ja/doc/casestudy/raspberrypi_mouse/raspimouse_tutorial_rtm_seminar/tutorial_rtm_seminar_ubuntu_part2)



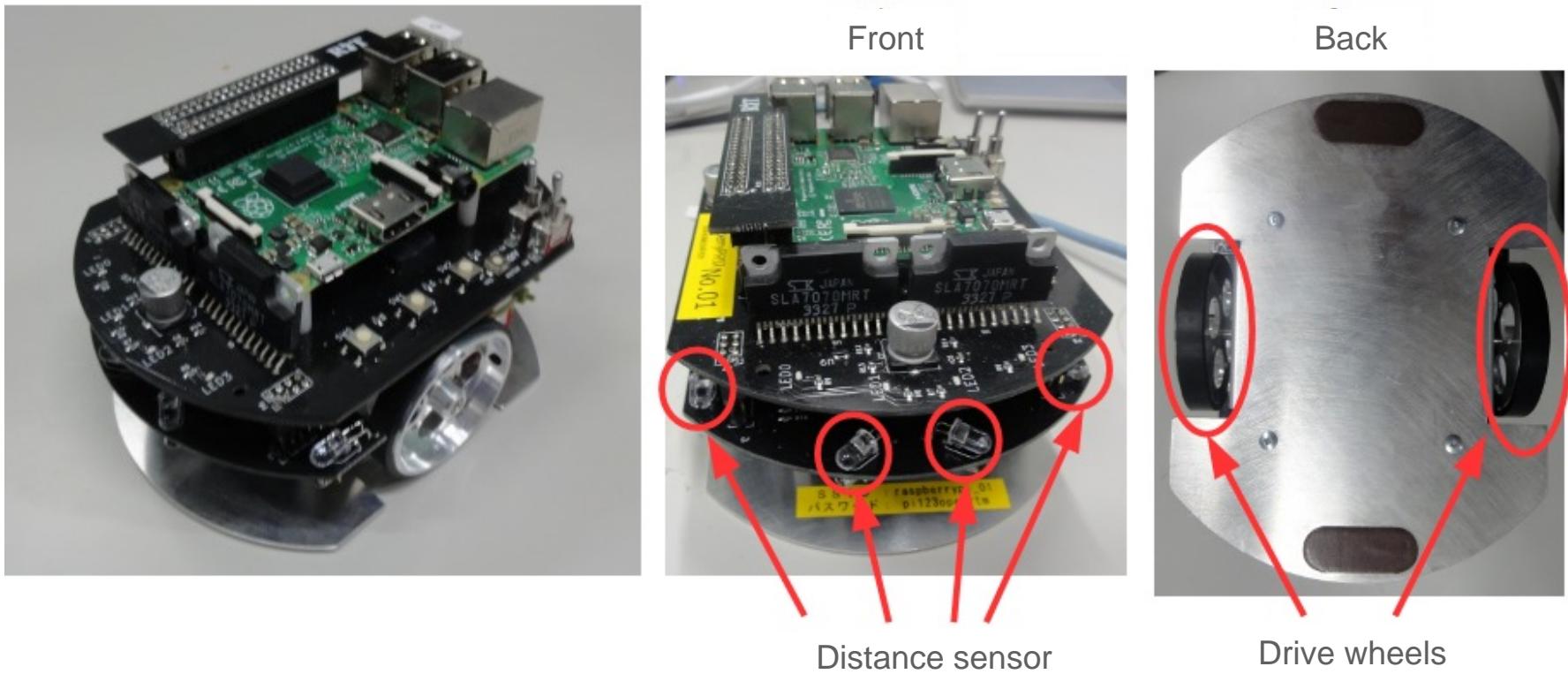
# Outline of the training

- Creating component to operate the mobile robot (Raspberry Pi Mouse) on the simulator.
  - Target speed input by GUI
  - Stops when the sensor value exceeds a certain level



# Raspberry Pi Mouse Overview

- Raspberry Pi Mouse is an two-wheel drive mobile robot sold by RT.



# The procedure of the training

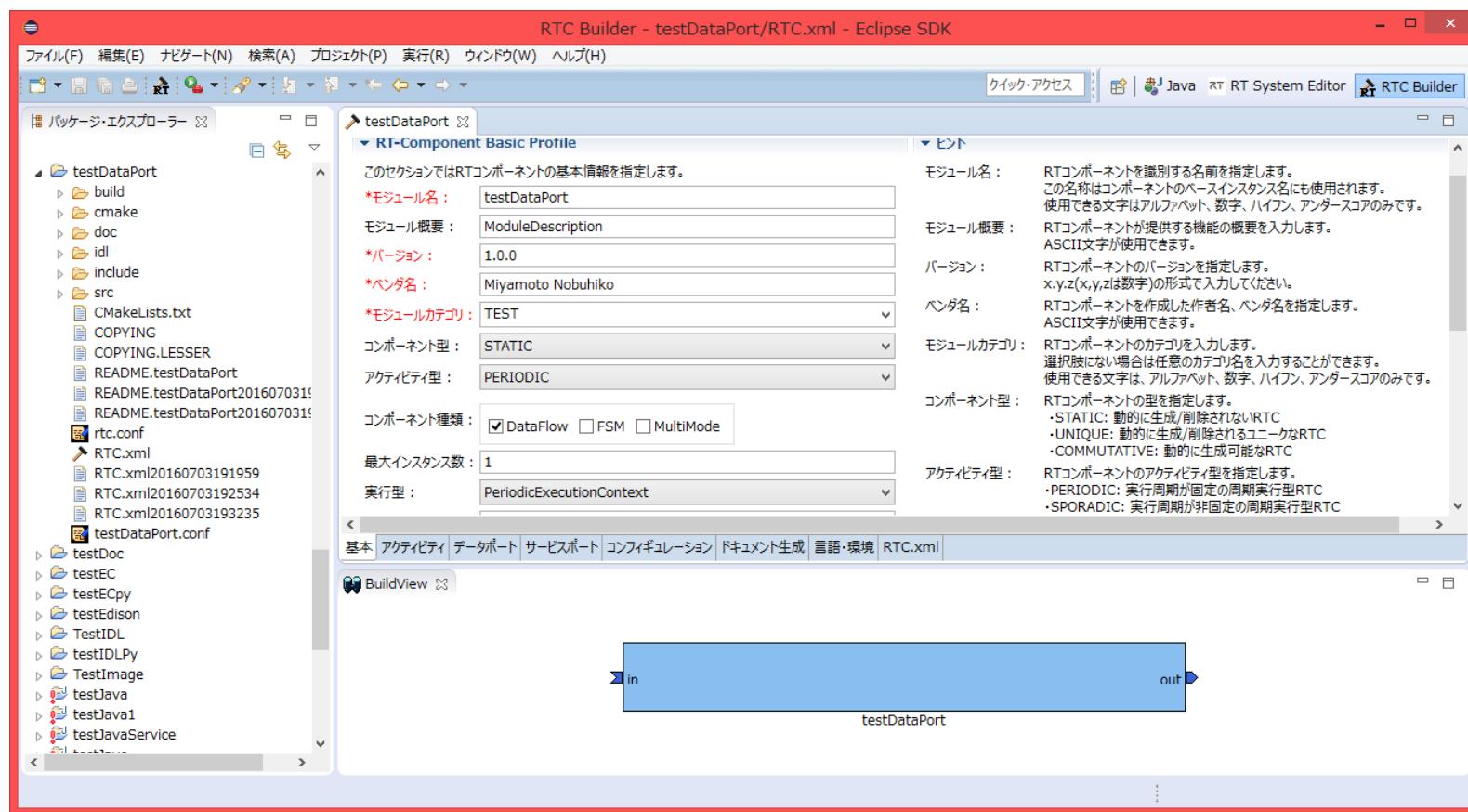
- Creating a template for source code using RTC Builder
- Edit source code and build.
  - Generate various files required for build
    - Various files generated by CMake
  - Editing source code
    - Editing RobotController.h and RobotController.cpp
  - Build
    - Visual Studio, Code::Blocks
- RT System creation and operation check using RT System Editor
  - RT system creation
    - Data port connection, configuration parameter settings.

# Component development tools

## RTC Builder

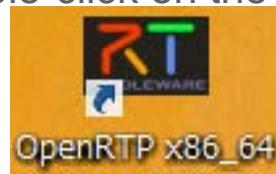
# RTC Builder

- A tool that enters component profile information and generates a template such as source code
  - Output C++, Python, Java, Lua source code



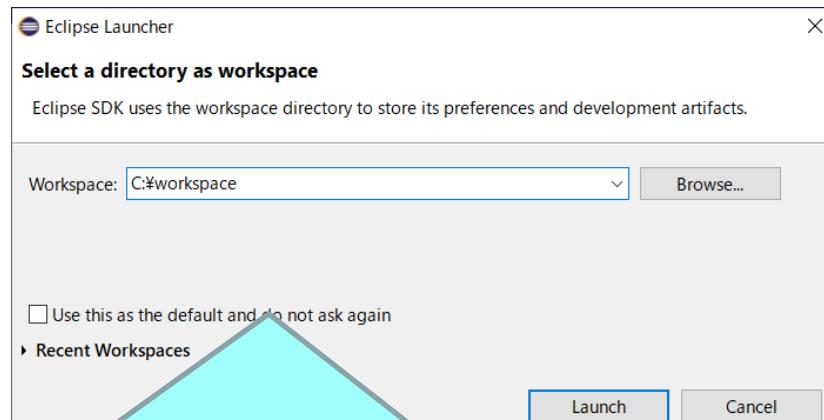
# Launch RTC Builder

- Steps to start RTC Builder
  - Windows(OpenRTM-aist 1.2)
    - Double-click on the desktop shortcut



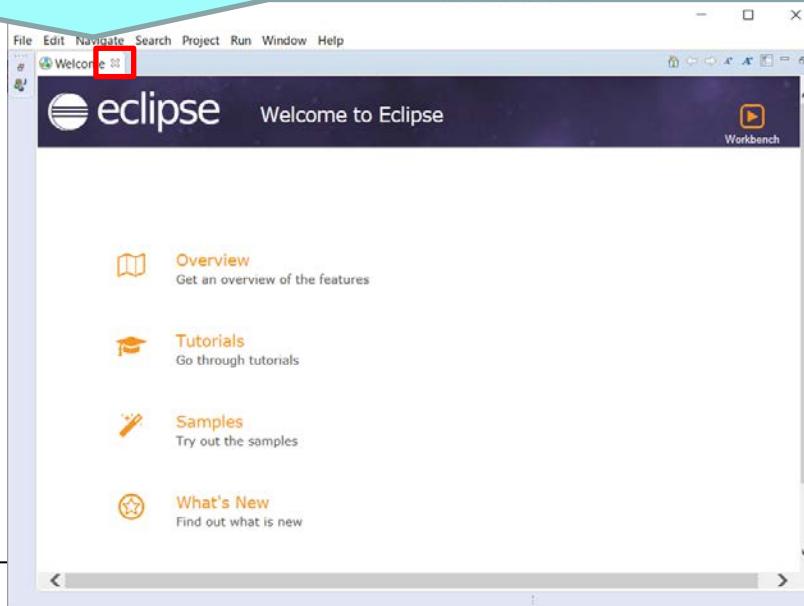
- Ubuntu
  - Enter the following command
  - \$ openrtp

# Start RTC Builder

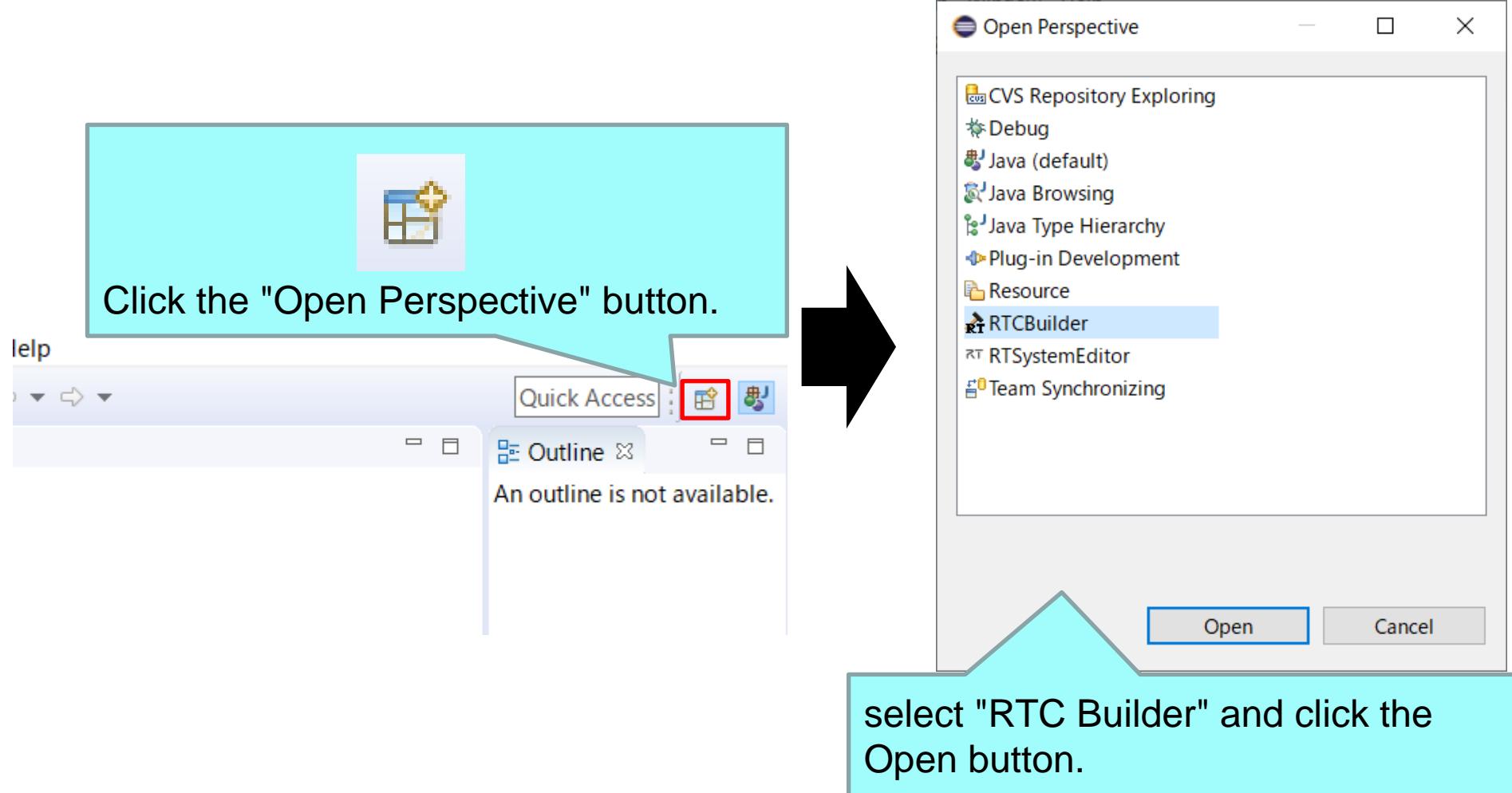


Set any location in the workspace and click Launch.

The Welcome page opens when you first start OpenRTP, so close it.

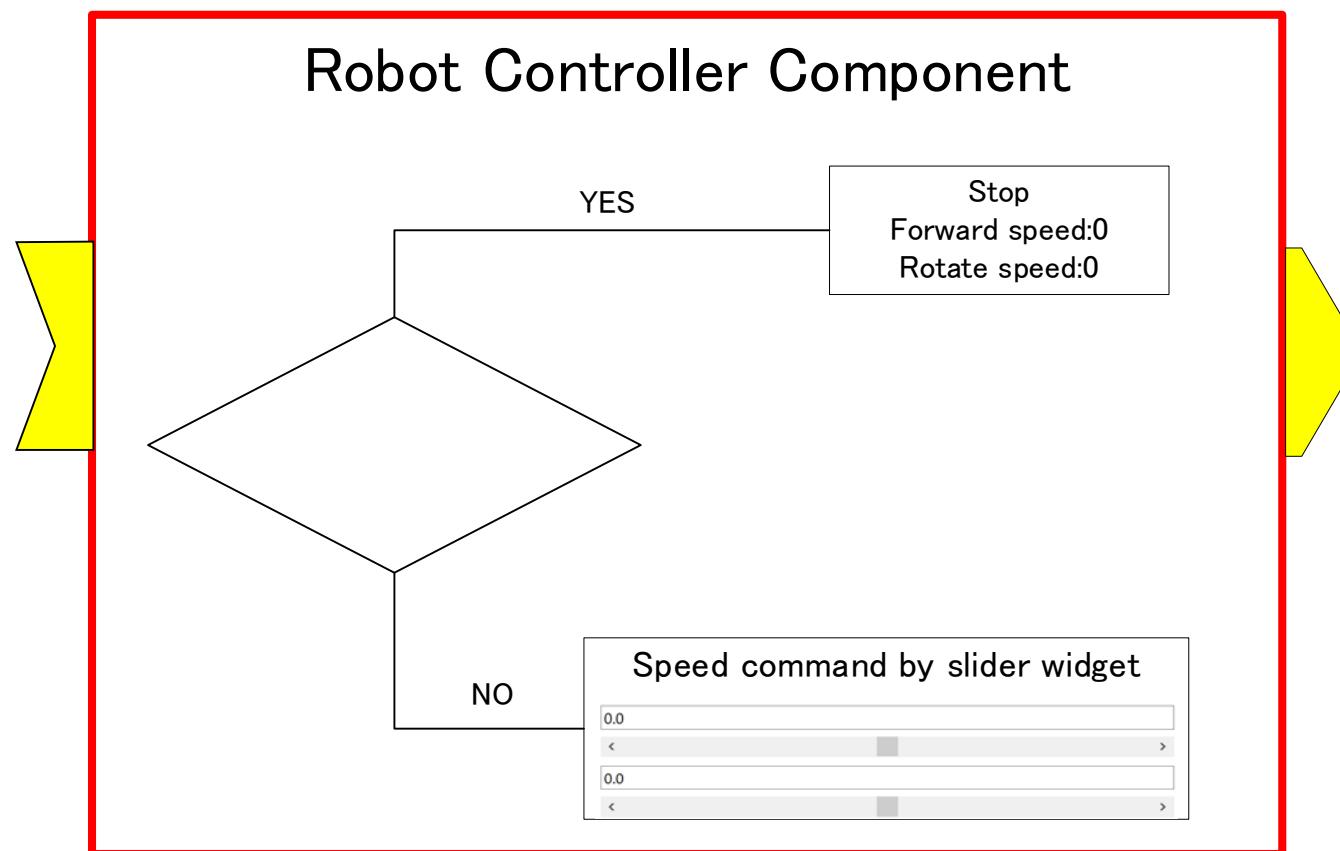


# Launch RTC Builder

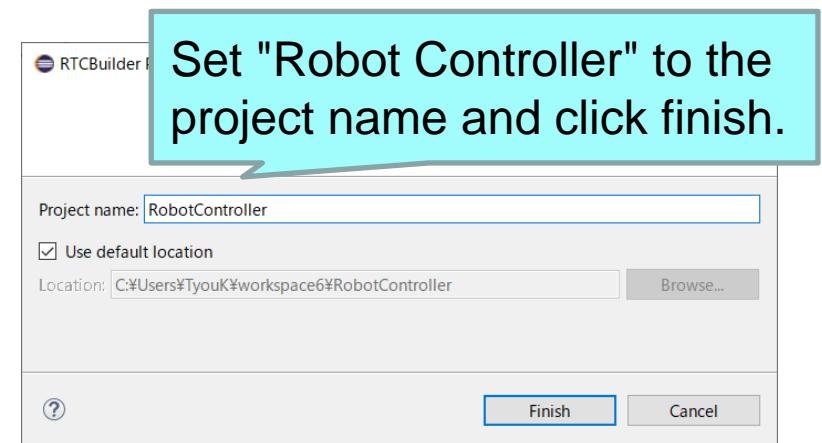
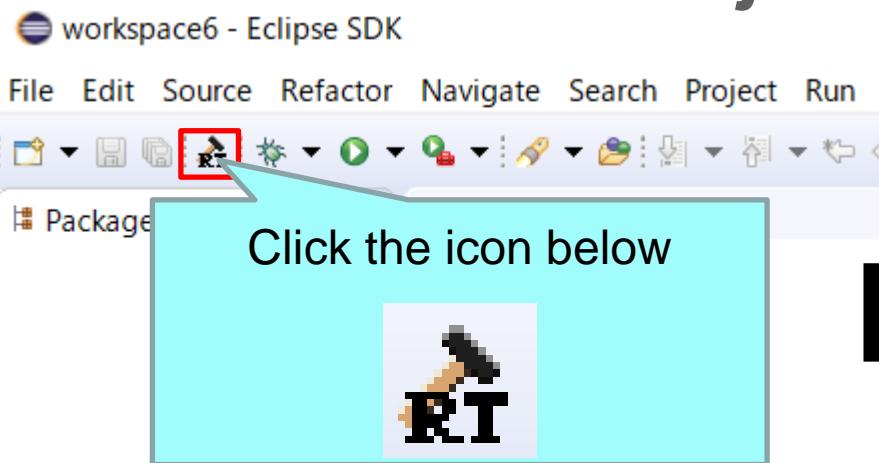


# Project creation

- Create skeleton code for the RobotController component.
  - Wheel type mobile robot operation component
    - Operate the robot with GUI
    - Stops when the sensor value exceeds a certain level



# Project creation



- A folder called "RobotController" is created in the directory specified in the workspace when Eclipse starts.
  - At this point, only "RTC.xml" and ".project" are generated
- The following items will be set
  - Basic profile
  - Activity profile
  - Data port profile
  - Service port profile
  - configuration
  - document
  - Language environment
  - RTC.xml

# Enter basic profile

- Set basic component information such as RT component profile information.
- Code generation, import/export, packaging process

\*RobotController × Basic ▾ RT-Component Basic Profile

RTC profile editor  
Enter each item here

This section defines RT-Component Basic information.

\*Component name : RobotController

Description : ModuleDescription

\*Version : 1.0.0

\*Vendor : VenderName

\*Category : Category

Component type : STATIC

Component's activity type : PERIODIC

Component kind :  DataFlow  ECU  MultiMode  Chaperone

Hint

Component name : Specifies the component name to This name will also be used as the base instance name of the component. Only alphabet, number, hyphen and underscore are available.

Description : Specifies summary of functions that can be performed by the component. ASCII is available.

Version : Specifies the RT-Component version. Please specify in x.y.z format (x,y,z).

Vendor : Specifies the RT-Component developer. ASCII is available.

Category : Specifies RT-Component category. If it is not one of the selections, another category can be specified. Only alphabet, number, hyphen and underscore are available.

Basic Activity Data Ports Service Ports Configuration Documentation Language and Environment RTC.xml

Select the Basic tab.

RobotController

# Enter basic profile

- Component name
  - RobotController
- Module overview
  - Optional (Robot Controller Component)
- version
  - Optional (1.0.0)
- Vendor name
  - Any
- Module category
  - Optional (Controller)
- Component type
  - STATIC
- Activity type
  - PERIODIC
- Component type
  - DataFlow
- Maximum number of instances
  - 1
- Execution type
  - PeriodicExecutionContext
- Execution cycle
  - 1000.0
- Overview
  - Any

**Basic**

▼ RT-Component Basic Profile

This section defines RT-Component Basic information.

*Component name :	RobotController
Description :	ModuleDescription
*Version :	1.0.0
*Vendor :	VenderName
*Category :	Category
Component type :	STATIC
Component's activity type :	PERIODIC
Component kind :	<input checked="" type="checkbox"/> DataFlow <input type="checkbox"/> FSM <input type="checkbox"/> MultiMode <input type="checkbox"/> Choreonoid
Number of maximum instance :	1
Execution type :	PeriodicExecutionContext
Execution rate :	1000.0
Abstract :	
RTC Type :	

# Activity settings

- Set the activity to use.

**Activity**

**Activity**

This section specifies the action callback that is used.

Component Action concerning the component's initialization and finalization		
onInitialize	onFinalize	
Component Action concerning the ExecutionContext's startup and shutdown		
onStartup	onShutdown	
Component Action in the alive state		
onActivated	onDeactivated	onAborting
onError	onReset	
Dataflow Component Action		
onExecute	onStateUpdate	onRateChanged
FSM Component Action		
onAction		
Mode Component Action		

**Basic Activity**

Select the Activity tab.

- Steps to enable specified activity

1. Select the activity name you want to use or not use  
(Displayed in red when selected)

Component Action in the alive state		
onActivated	onDeactivated	onAborting
onError	onReset	
Dataflow Component Action		
onExecute	onStateUpdate	onRateChanged
FSM Component Action		
onAction		
Mode Component Action		

2. After selecting the activity name,  
select ON-OFF.

Activity name :

ON  OFF

Activated activity names have a blue background

**\*RobotController**

Component Action in the alive state		
onStartup	onShutdown	
onActivated	onDeactivated	onAborting
onError	onReset	
Dataflow Component Action		
onExecute	onStateUpdate	onRateChanged
FSM Component Action		
onAction		
Mode Component Action		
onModeChanged		

**Documentation**

This section specifies a short description of each action.  
If the action above is selected, each document can be described.

Activity name :

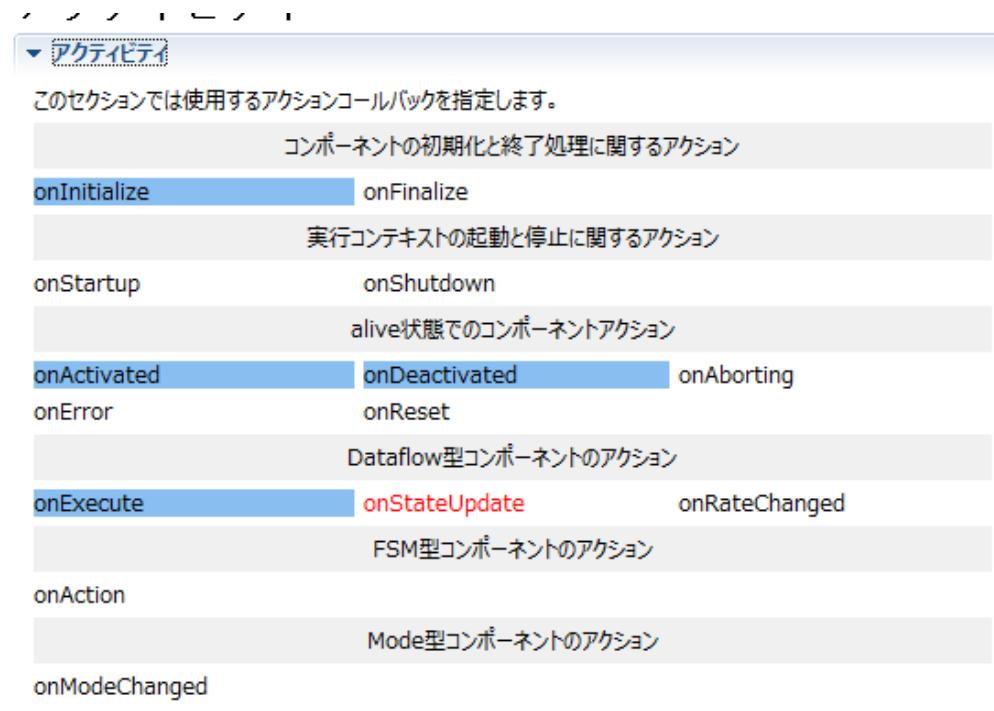
ON  OFF

# Activity settings

コールバック関数	処理
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

The screenshot shows the 'Data Ports' configuration window for a 'RobotController' component. The 'DataPort Profile' tab is selected. It displays two tables for adding ports: one for 'Port Name (DataInPort)' and another for 'Port Name (DataOutPort)'. Below these tables is a 'Detail' section with fields for 'Port name', 'Data Type', 'IDL file', and 'Var Name'. A large green callout box points to the 'DataPort Profile' tab with the text 'Select the Data Port tab.'

- データポートを追加する手順

## Data Ports

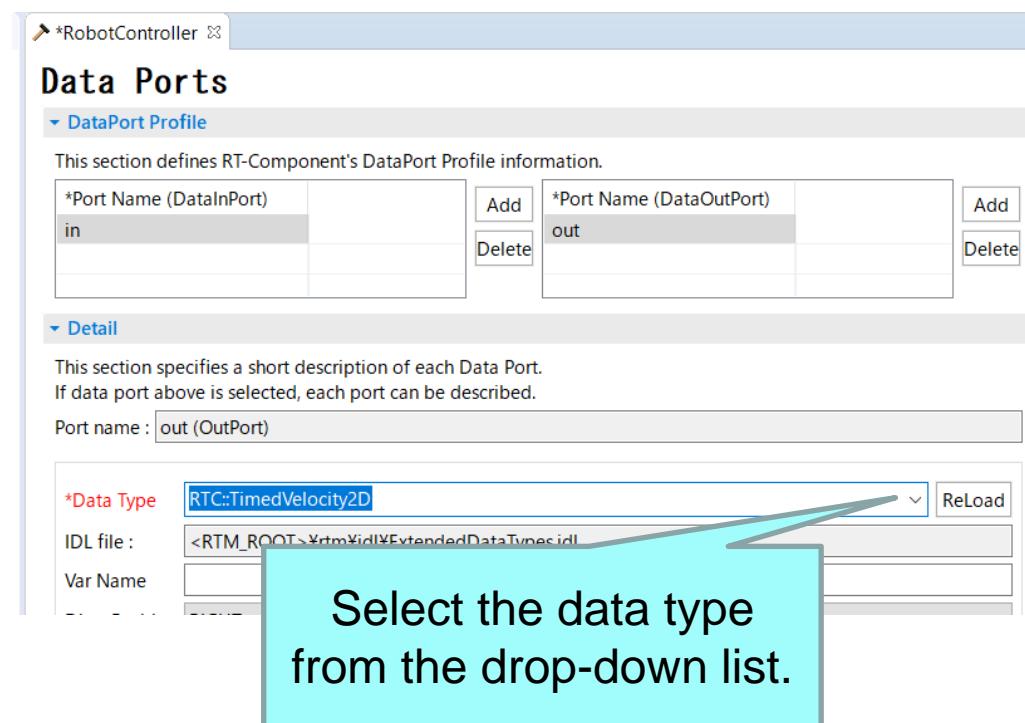
This screenshot shows the same 'Data Ports' configuration window. A large green callout box highlights the 'Add' button under the 'Port Name (DataInPort)' table with the text 'Click the add button for ImPort or OutPort.' Another green callout box highlights the 'in' entry in the 'Port Name (DataInPort)' table with the text 'Click the port name to rename it.' A large black arrow points from the first step to the second. The 'Detail' section at the bottom is also visible.

## Data Ports

This screenshot shows the 'Data Ports' configuration window after port creation. The 'Port Name (DataInPort)' table now contains 'in'. A large green callout box highlights the 'in' entry with the text 'Set each item.' The 'Detail' section at the bottom is also visible.

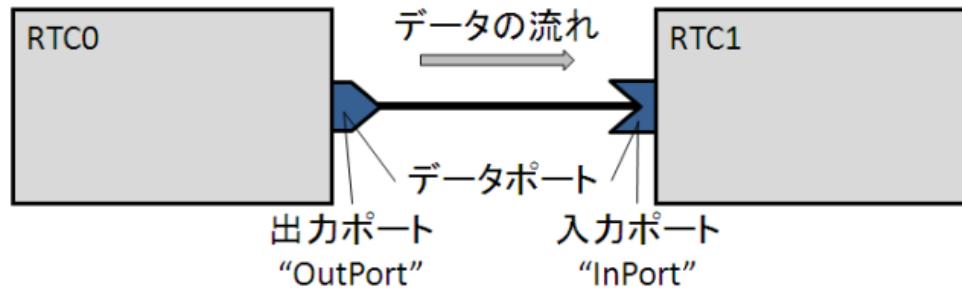
# Data port settings

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

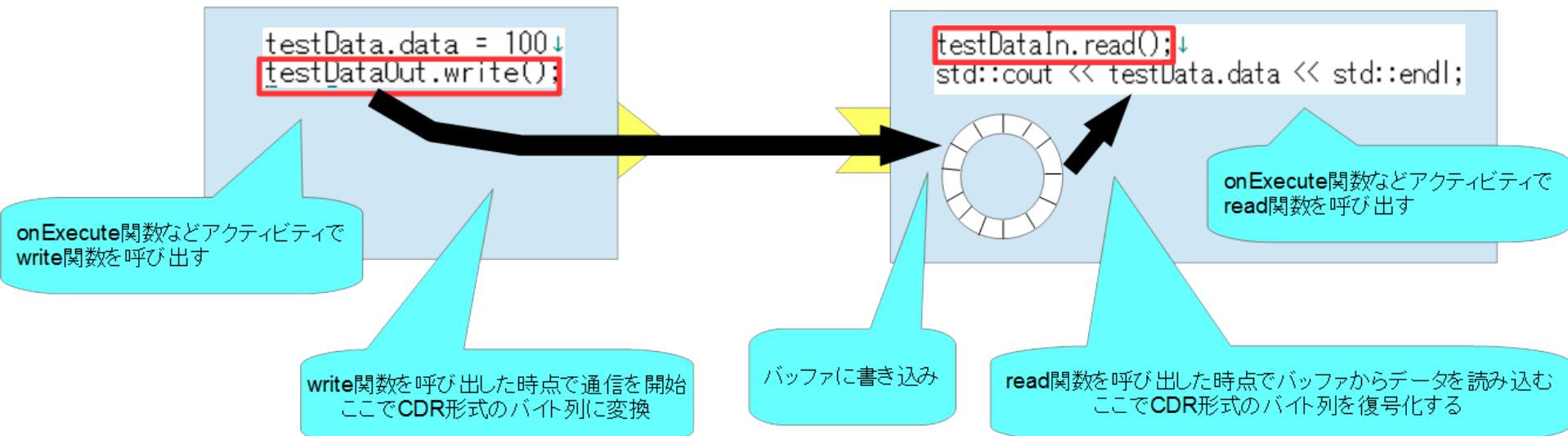


# About the data port

- Port for communicating continuous data

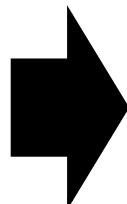


- 以下の例はデータフロー型がpush、サブスクリプション型がflush、インターフェース型がcorba\_cdrの場合



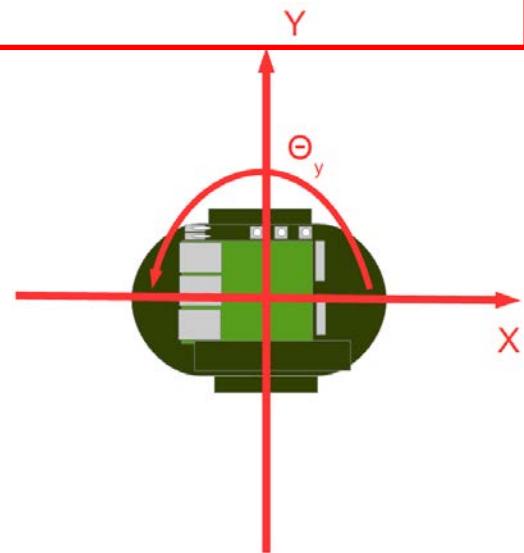
# About RTC::TimedVelocity2D type

- Data type for expressing the speed of a mobile robot defined in ExtendedDataTypes.idl.
  - **vx**: Velocity in the X-axis direction
  - **vy**: Velocity in the Y-axis direction (0 assuming the wheels do not skid)
  - **va**: Angular velocity around the Z axis



**Set straight speed with vx and rotation speed with va**

```
struct Velocity2D{  
    /// Velocity along the x axis in metres per second.  
    double vx;  
    /// Velocity along the y axis in metres per second.  
    double vy;  
    /// Yaw velocity in radians per second.  
    double va;  
};
```



# Configuration Parameter settings

- Add and set configuration parameters

RT-Component Configuration Parameter

This section defines RT-Component Configuration Parameter.

\*Name

Add  Delete

Hint  Config.

Parameter

Detail  Data type  Default

This section specifies each Configuration Parameter description.

Parameter name :

\*Type  Variable

\*Default Value

Variable name :

Basic Activity Data Ports Service Ports Configuration Documentation Language and Environment

Select the Configuration tab.

- コンフィギュレーションパラメータを追加する手順

RT-Component Configuration Parameter

This section defines RT-Component Configuration Parameter.

\*Name

Add  Delete

Detail

This section specifies each Configuration Parameter

Parameter name :

\*Type

Click the add button.

## RT-Component Configuration Parameter

This section defines RT-Component Configuration Parameter.

\*Name  speed\_x

Add  Delete

Detail

This section specifies each Configuration Parameter description.

Parameter name :  speed\_x

\*Type

Set each item.

# Configuration Parameter settings

- Set the following configuration parameters
  - speed\_x**
    - Type : double
    - Default Value : 0.0
    - Constraint :  $-1.5 < x < 1.5$
    - Widget : slider
    - Step: 0.01
  - speed\_r**
    - Type : double
    - Default Value : 0.0
    - Constraint :  $-2.0 < x < 2.0$
    - Widget : slider
    - Step: 0.01

This section defines RT-Component Configuration Parameter.

*Name		Add
speed_x		
speed_r		
stop_d		

**Detail**

This section specifies each Configuration Parameter description.

Parameter name : **speed\_x**

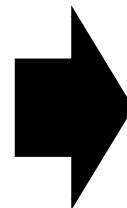
*Type	double
*Default Value	0.0
Variable name :	
Unit :	
Constraint :	$-1.5 < x < 1.5$
Widget :	slider
Step :	0.01

Enable operation of mobile robots using GUI (slider)

0.0	
0.1	

# Configuration parameter constraints, and widget settings

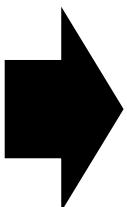
- Show GUI when editing configuration parameters in RT System Editor

A simple text input field containing the number "0".

- Widget:text

A text input field containing the number "40", with up and down arrow buttons on the right side.

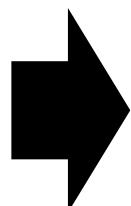
- Constraint:  $0 \leq x \leq 100$
- Widget: spin
- Step : 10

A horizontal slider control with a central gray bar and arrows at both ends. The number "40" is displayed in the text input field above it.

- Constraint:  $0 \leq x \leq 100$
- Widget: slider
- Step : 10

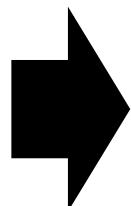
# Configuration parameter constraints, and widget settings

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



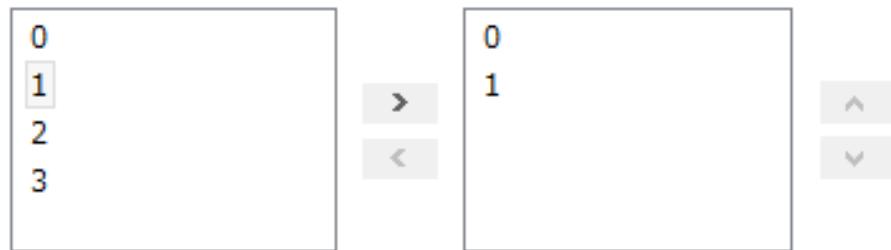
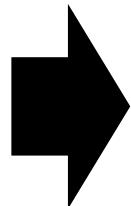
A horizontal row of four radio buttons. The first two are labeled '0' and '3' respectively, each with an empty circle. The third and fourth buttons are labeled '1' and '2', with the '2' button having a filled circle, indicating it is selected.

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



A horizontal row of three checkboxes. The first two are labeled '0' and '3' respectively, with the '0' checkbox having a checked mark. The third checkbox is labeled '1' and has an empty square, indicating it is not selected. The fourth checkbox is labeled '2' and has a checked mark, indicating it is selected.

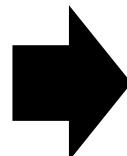
- Constraint : (0,1,2,3)
- Widget : ordered\_list



A configuration interface for an ordered list. On the left, a vertical list shows items 0, 1, 2, and 3. To the right, there are two rectangular boxes. The left box contains items 0 and 1. Between the boxes are two small grey buttons with arrows: one pointing right and one pointing left. The right box is currently empty. On the far right, there are two additional small grey buttons with arrows, one pointing up and one pointing down.

# Configuration Parameter settings

- Set the following configuration parameters
  - stop\_d**
    - Type : int
    - Default Value : 30



Stop when the sensor value is greater than or equal to this value

This section defines RT-Component Configuration Parameter.

*Name	Add
speed_x	
speed_r	
stop_d	

▼ Detail

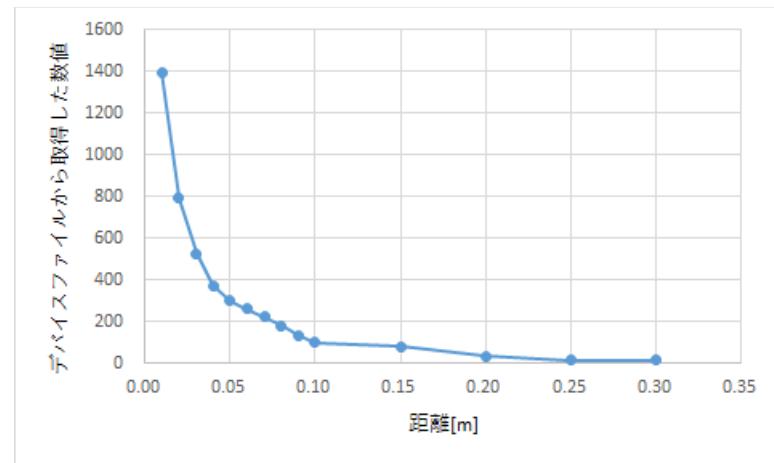
This section specifies each Configuration Parameter description.

Parameter name : stop\_d

*Type	int
*Default Value	30
Variable name :	
Unit :	
Constraint :	
Widget :	text
Step :	

# Raspberry Pi mouse distance sensor

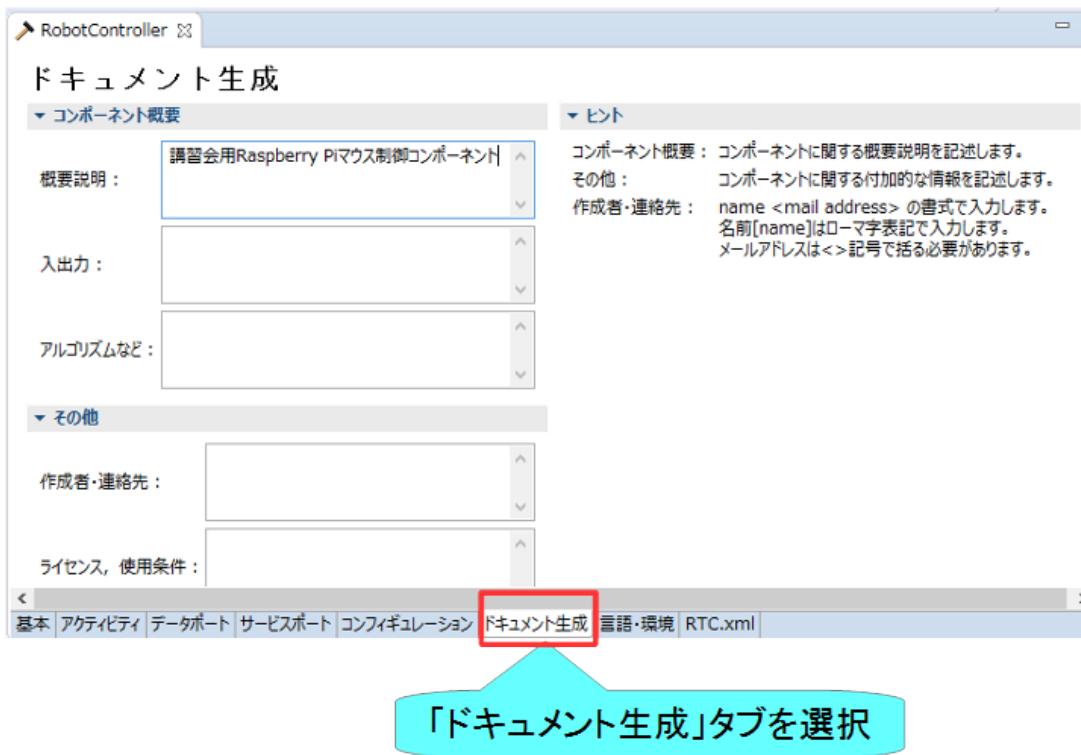
- Raspberry Pi mouse is equipped with a distance sensor
  - The measured value becomes larger as the distance to the object is shorter.



- The simulator also calculates and outputs a value close to this data.

# ドキュメントの設定

- 各種ドキュメント情報を設定



- 今回は適当に設定しておいてください。
  - 空白でも大丈夫です

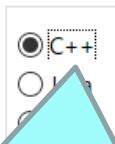
# Language setting

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

## Language and Environment

### Language

This section defines a language that is used.



Set the language.  
This time, set "C ++".

### Hint

Language : Si  
Environment : Si  
SL  
TI  
is

This section defines depending libraries and Oses etc that are used.

Version	OS		Add	Delete

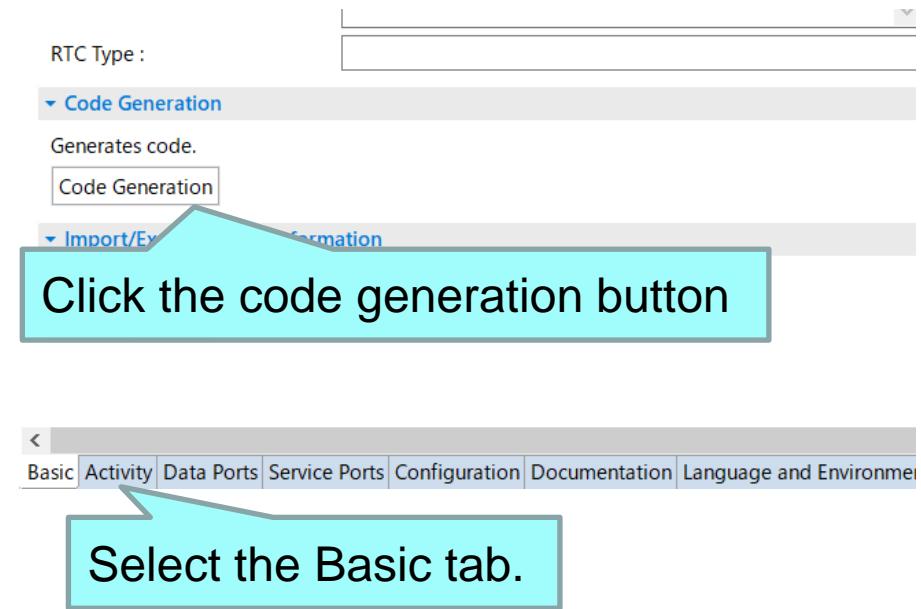
Detail information

OS Version		Add	Delete

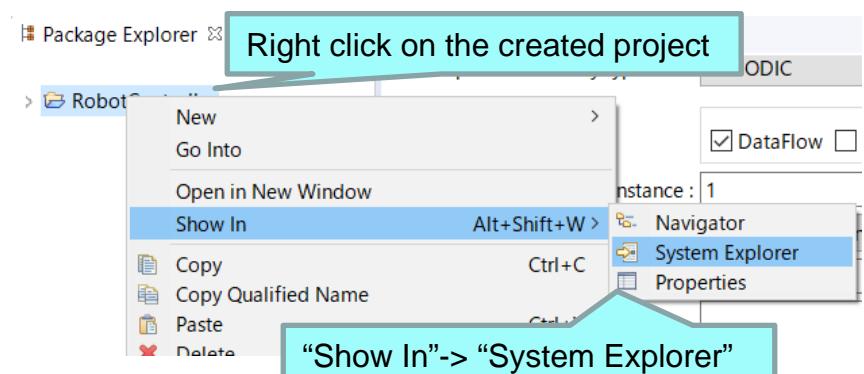
Select the Language and Environment tab.

# 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

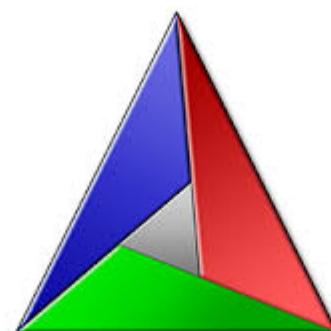
- Generate various files required for build
  - Various files generated by CMake
- Edit the source code
  - Edit RobotController.h
  - Edit RobotController.cpp
- Build
  - Windows: Visual Studio
  - Ubuntu: Code::Blocks

# 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



CMakeLists.txt



CMake



Makefile

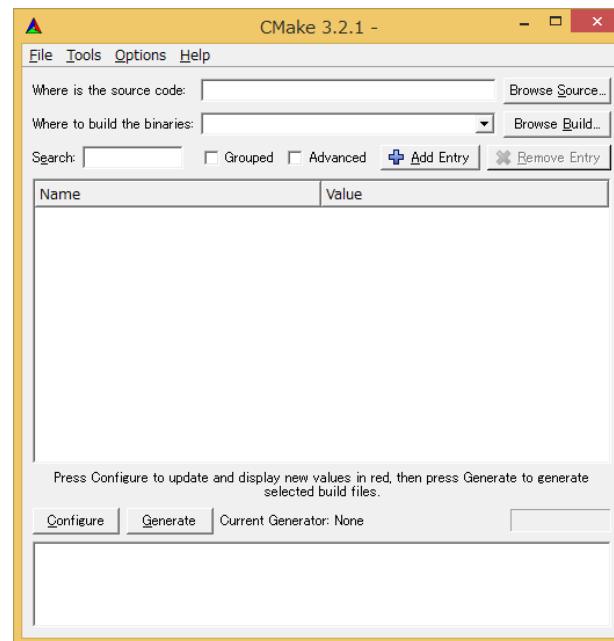


Visual Studio  
(ソリューションファイル、  
プロジェクトファイル等)

# Generating files needed for build

- Start CMake

- Windows 7
  - 「スタート」→「すべてのプログラム」→「CMake」→「CMake (cmake-gui)」
- Windows 8.1
  - 「スタート」→「アプリビュー(右下矢印)」→「CMake」→「CMake (cmake-gui)」
- Windows 10
  - 左下の「ここに入力して検索」にCMakeと入力して表示されたCMake(cmake-gui)を起動
- Ubuntu
  - Enter the following command
  - cmake-gui



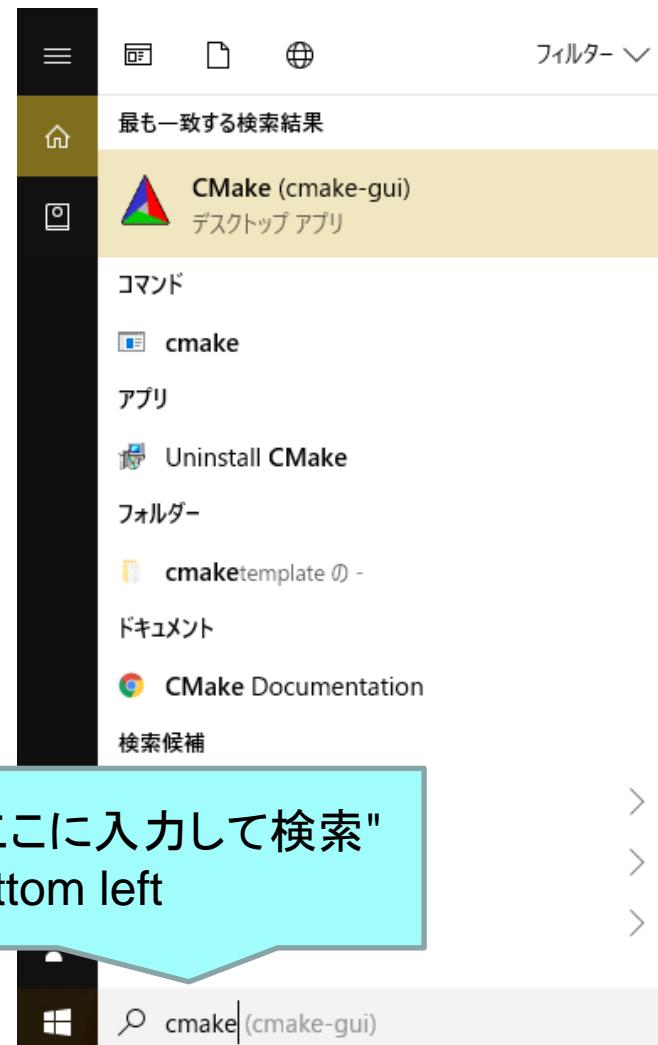
# cmake-guiの起動

- Windows 8.1



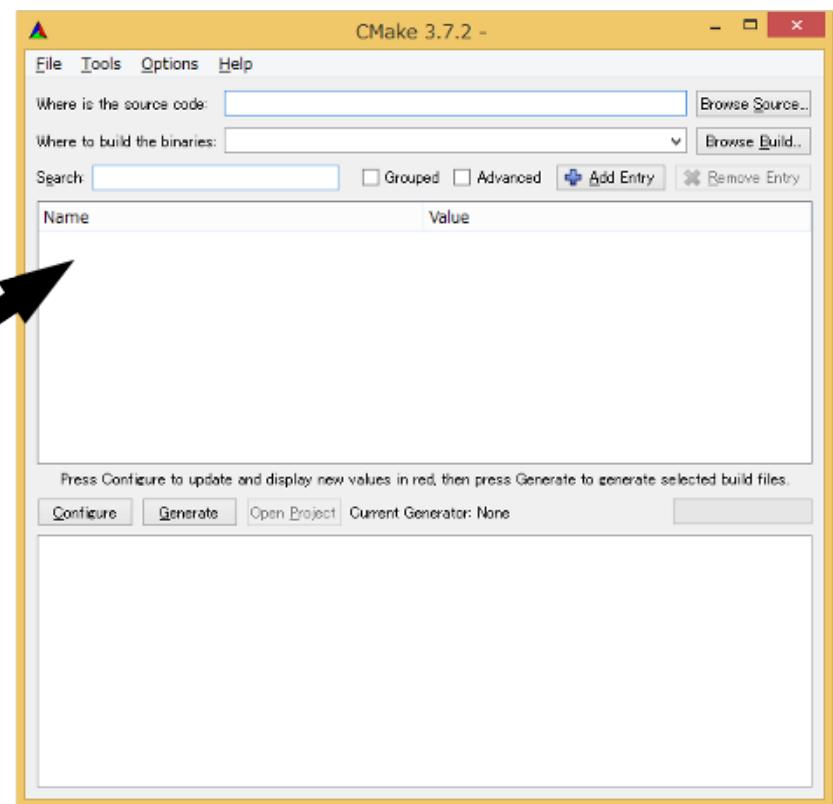
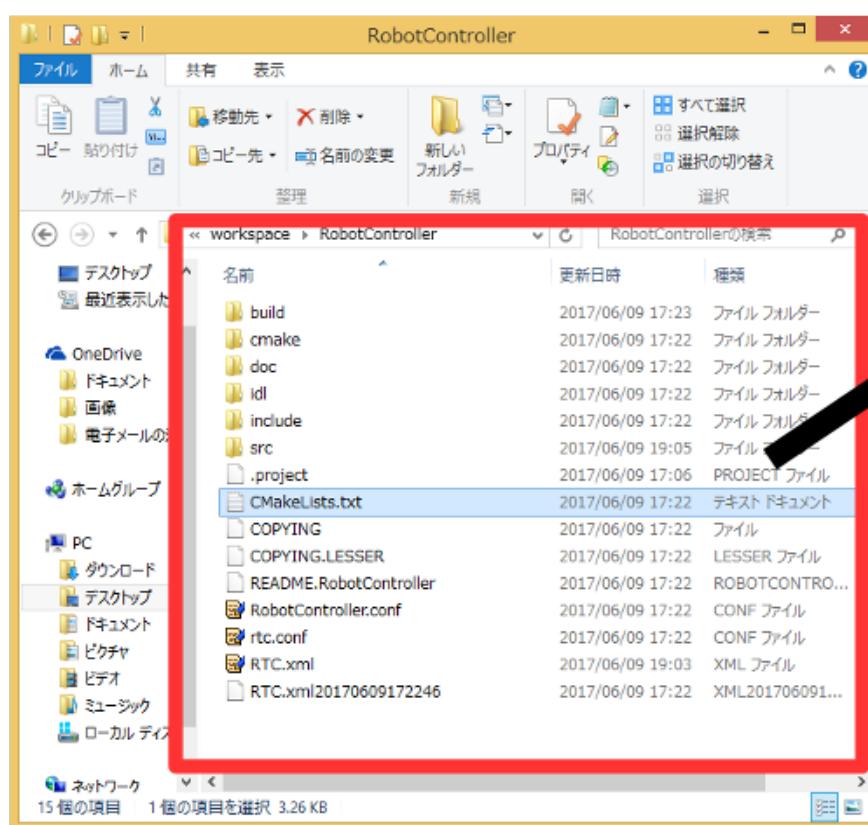
# cmake-guiの起動

- Windows 10

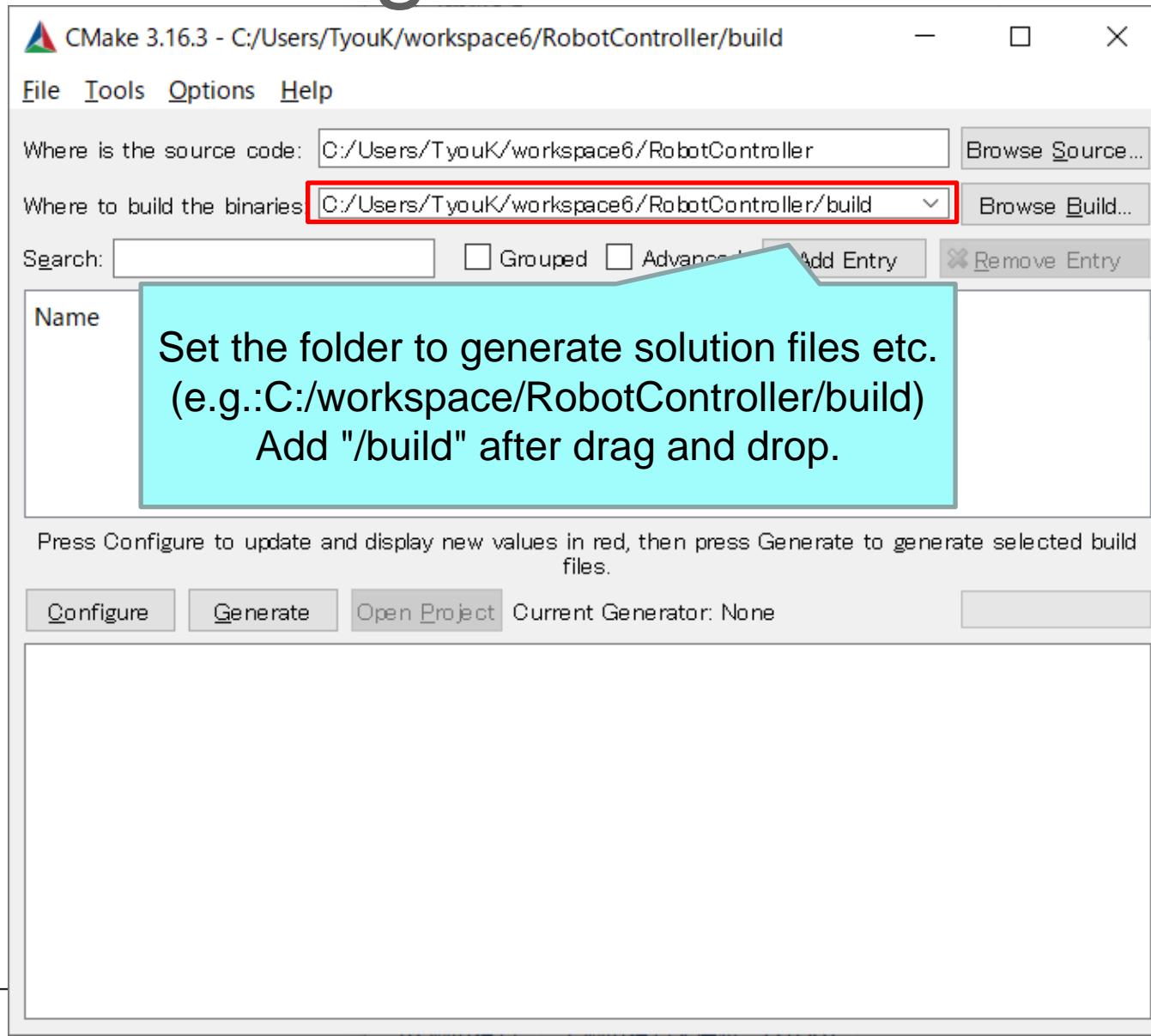


# Generating files needed for build

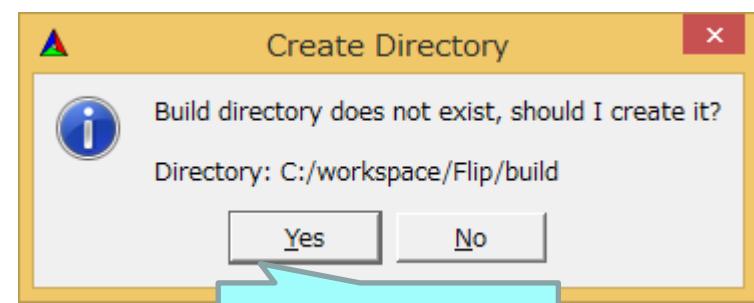
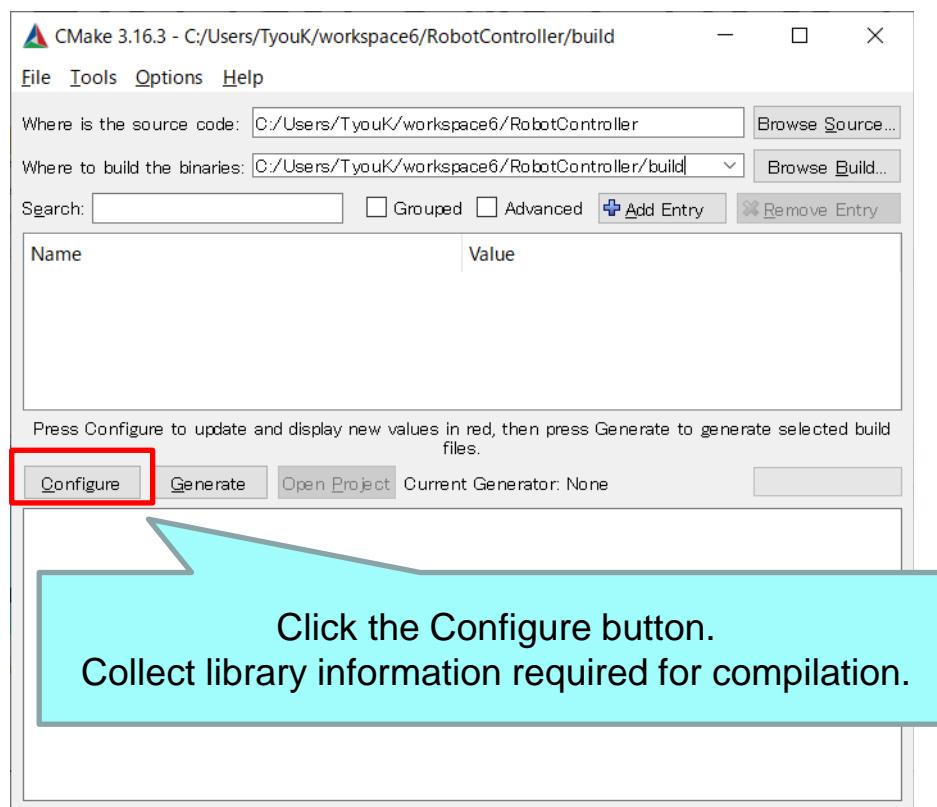
- Drag and drop CMakeLists.txt onto cmake-gui
  - CMakeLists.txt is a folder for projects generated by RTC Builder.  
(例：C:\workspace\RobotController)



# Generating files needed for build



# ビルドに必要なファイルの生成



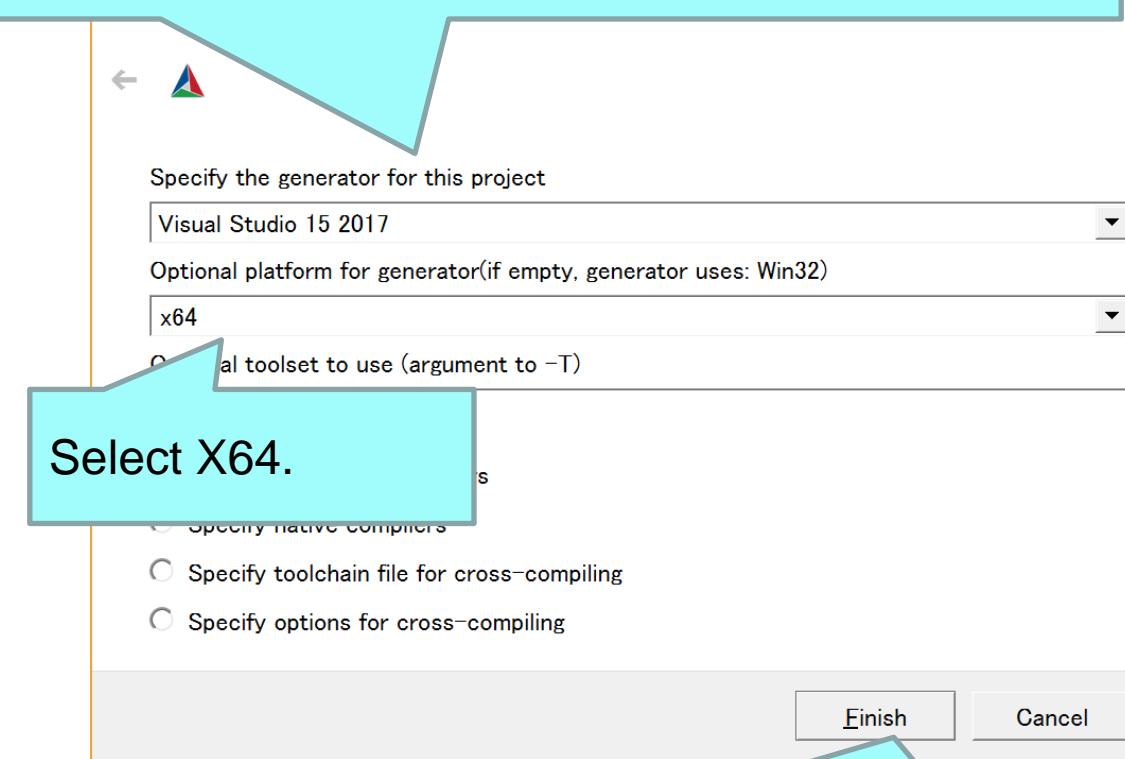
# For CMake 3.14 or later

Build environment settings

Visual Studio 2017 → Visual Studio 15 2017

Visual Studio 2013 → Visual Studio 12 2013

Code::Blocks → CodeBlocks-Unix Makefiles



After setting, press the Finish button.

# CMake 3.13以前の場合

## ビルド環境の設定

Visual Studio 2013 32bit → Visual Studio 12 2013

Visual Studio 2013 64bit → Visual Studio 12 2013 Win64

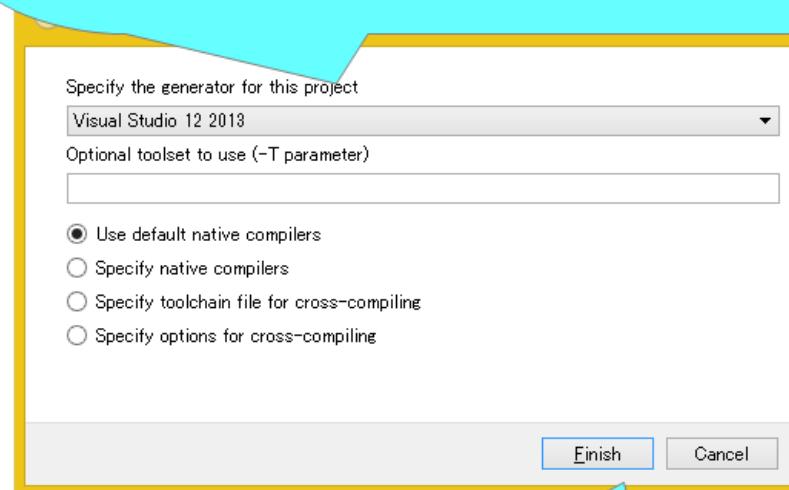
Visual Studio 2017 32bit → Visual Studio 15 2017

Visual Studio 2017 64bit → Visual Studio 15 2017 Win64

Code::Blocks → CodeBlocks-Unix Makefiles

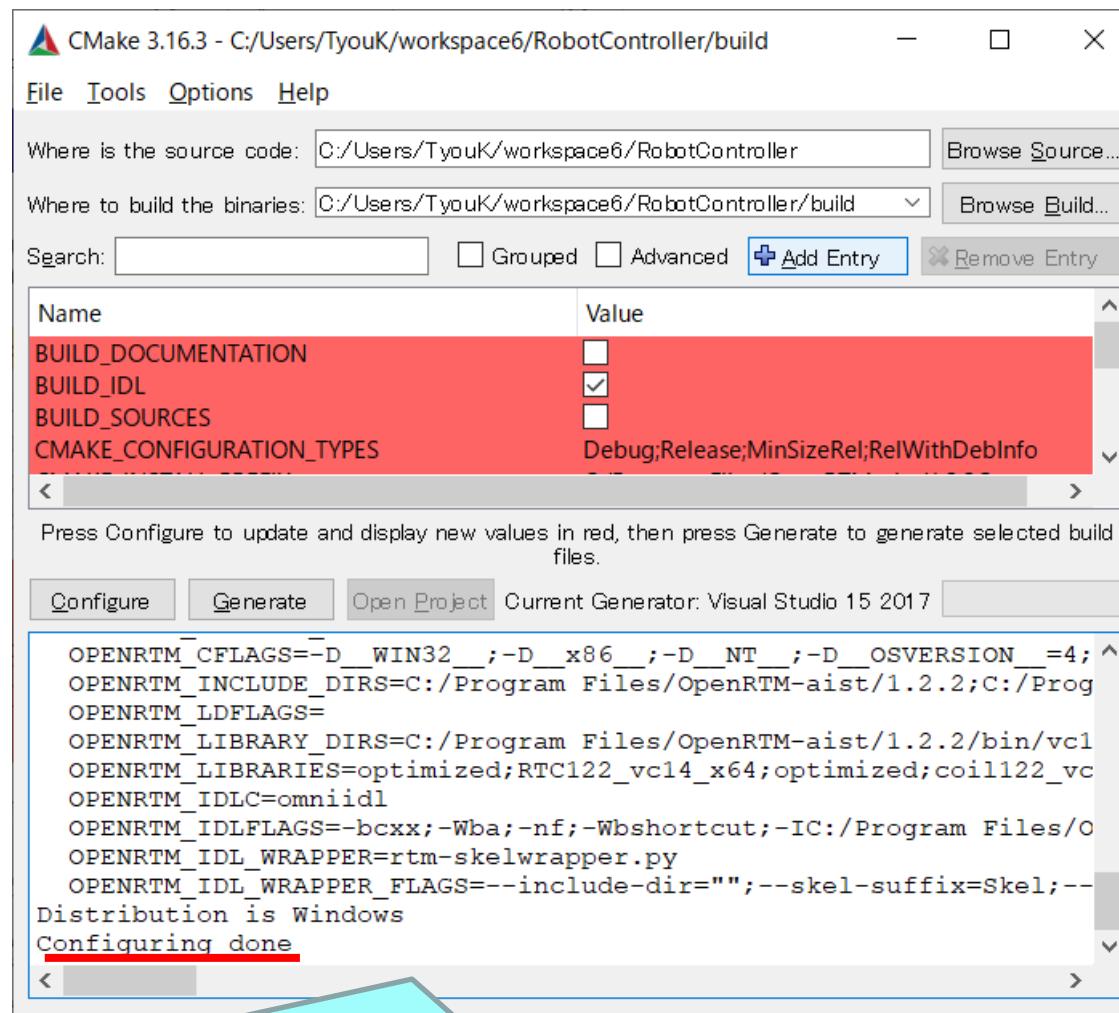
※32bitか64bitかはインストールした

OpenRTM-aistが32bitか64bitかで選択



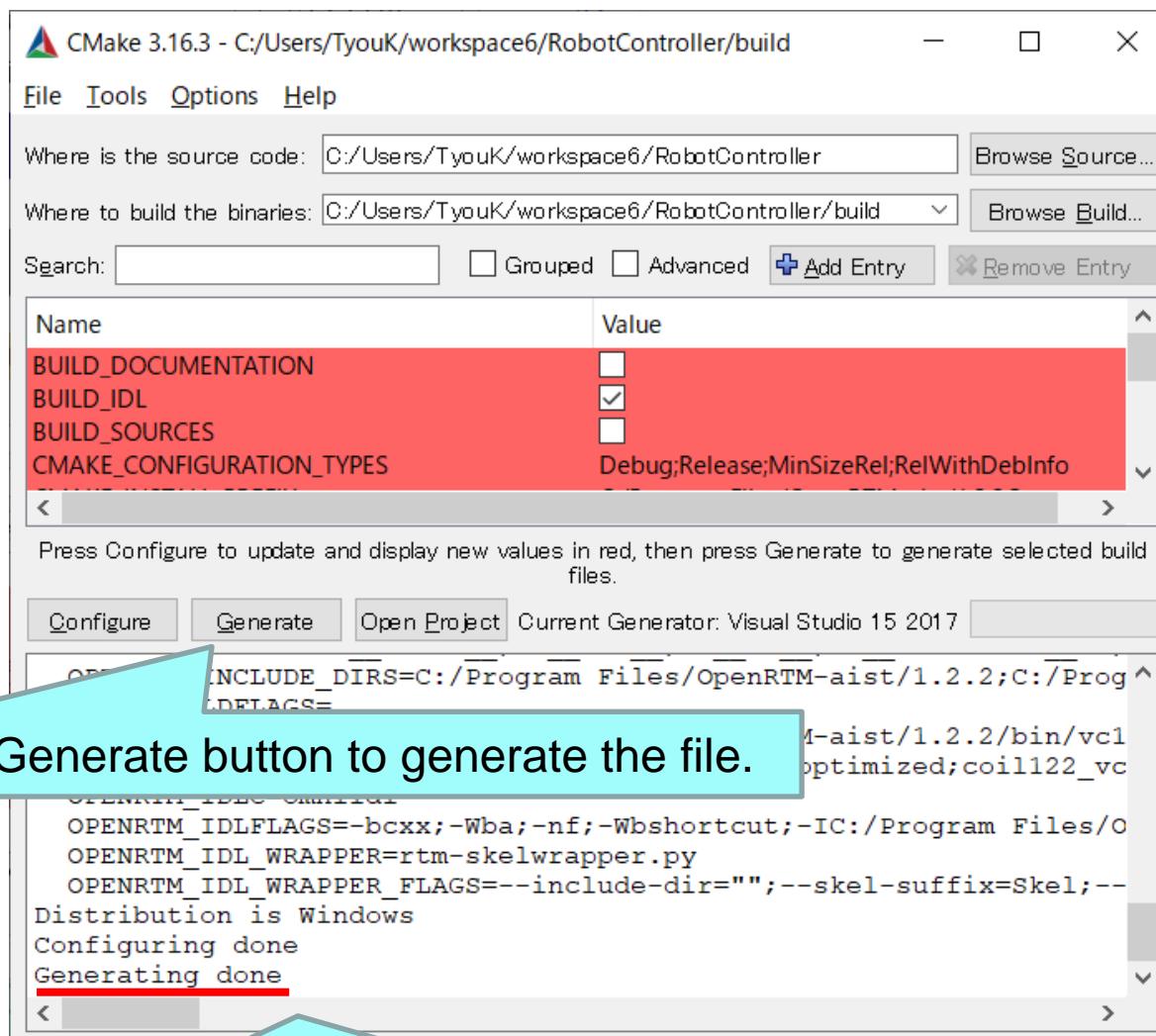
設定後、Finishボタンを押す

# Generating files needed for build



If "Configure done" is displayed, it is successful.

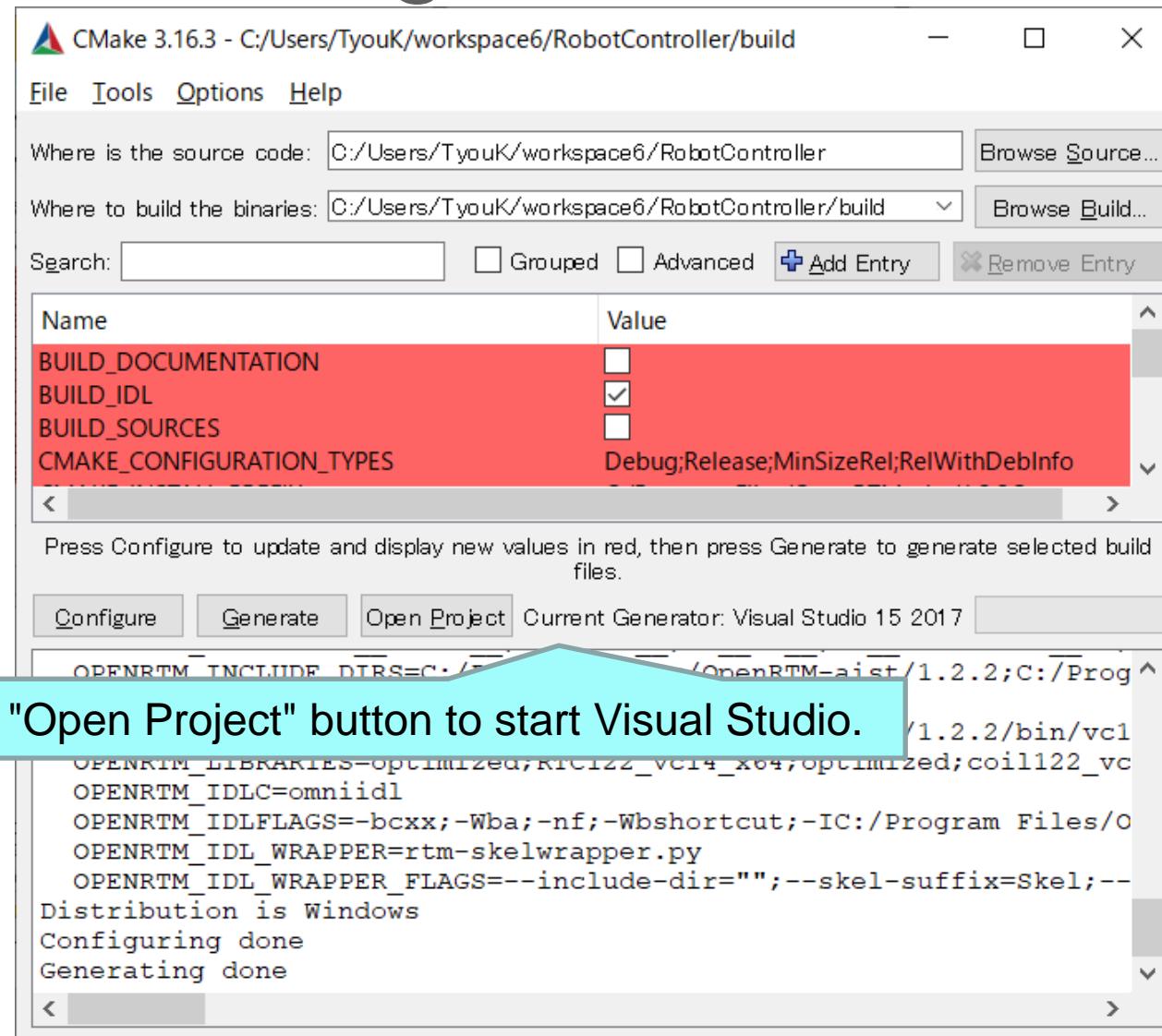
# Generating files needed for build



Click the Generate button to generate the file.

If "Generating done" is displayed, it is successful.

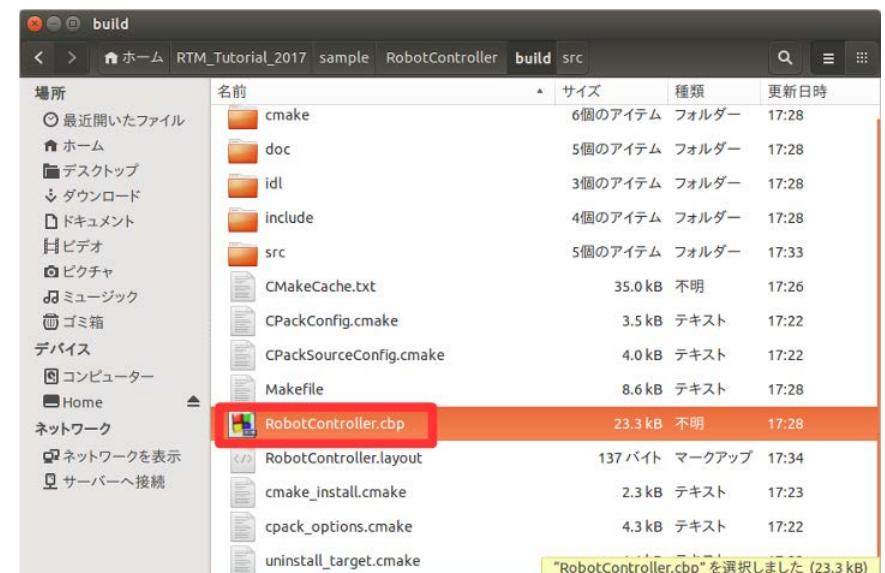
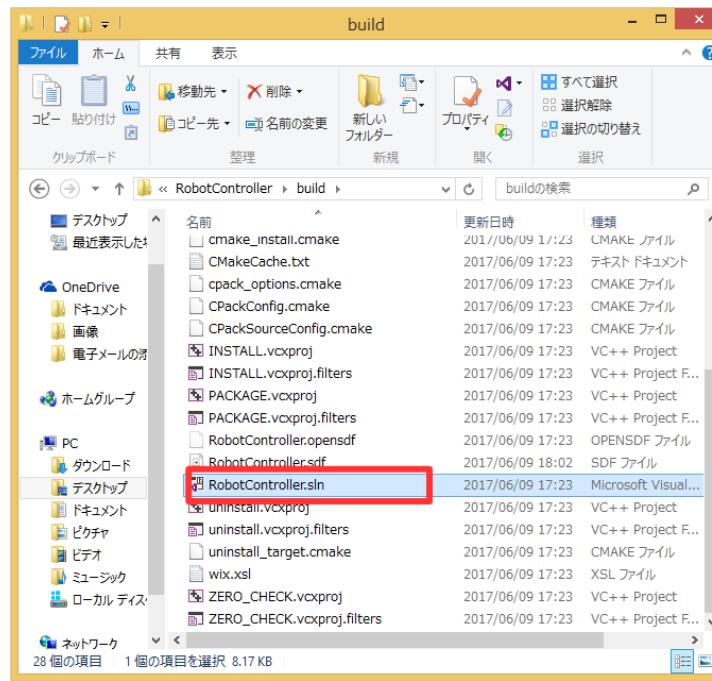
# Editing source code



Click the "Open Project" button to start Visual Studio.

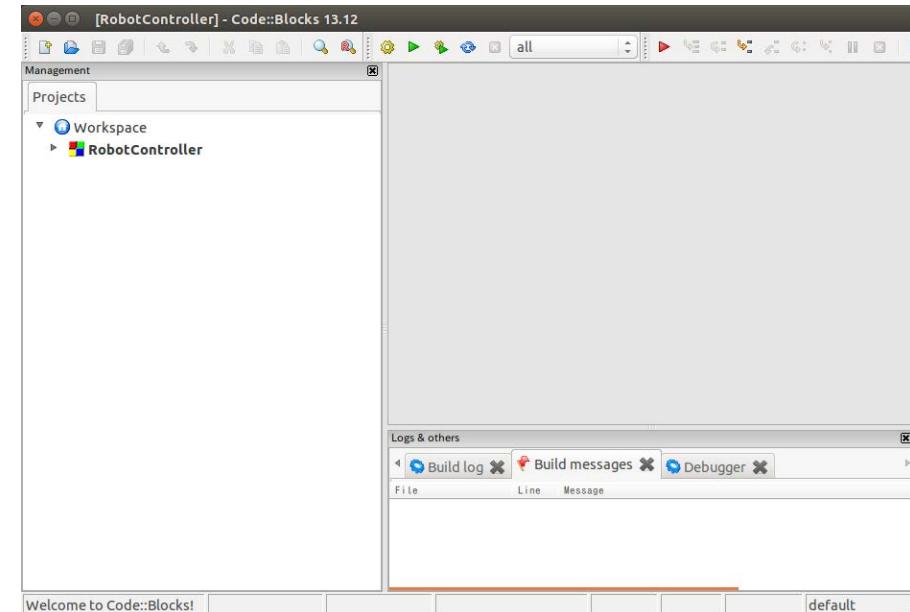
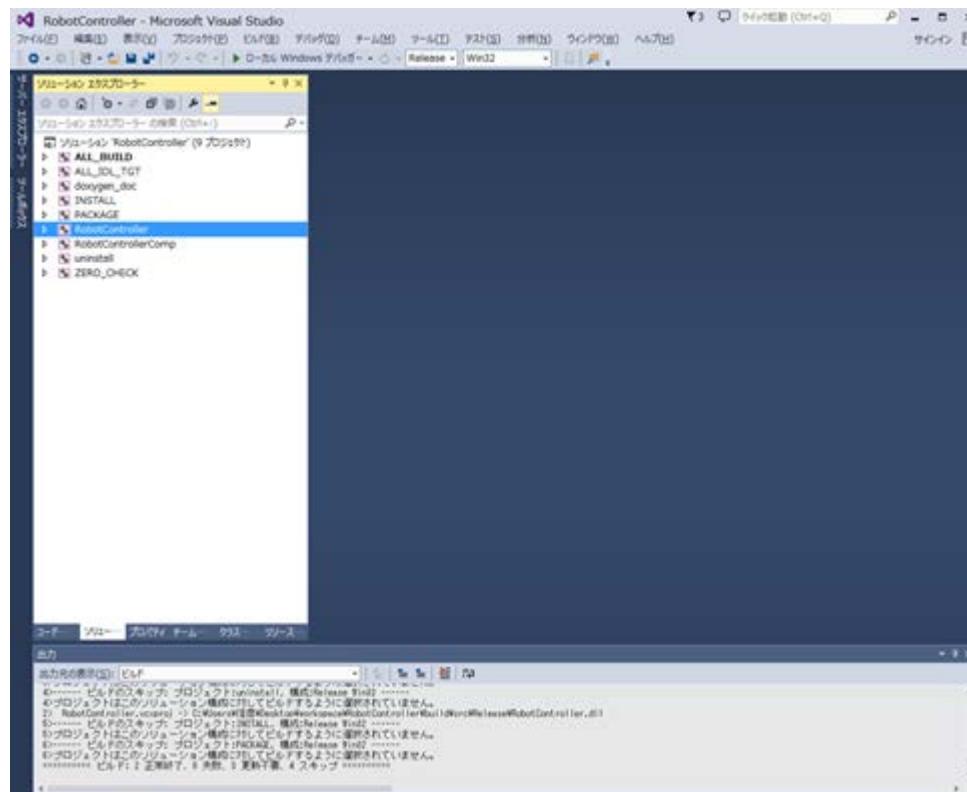
# Editing source code

- If the version of CMake-gui is old, there is no "Open Project" button, so double-click the file to open it.
  - Windows
    - Left-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

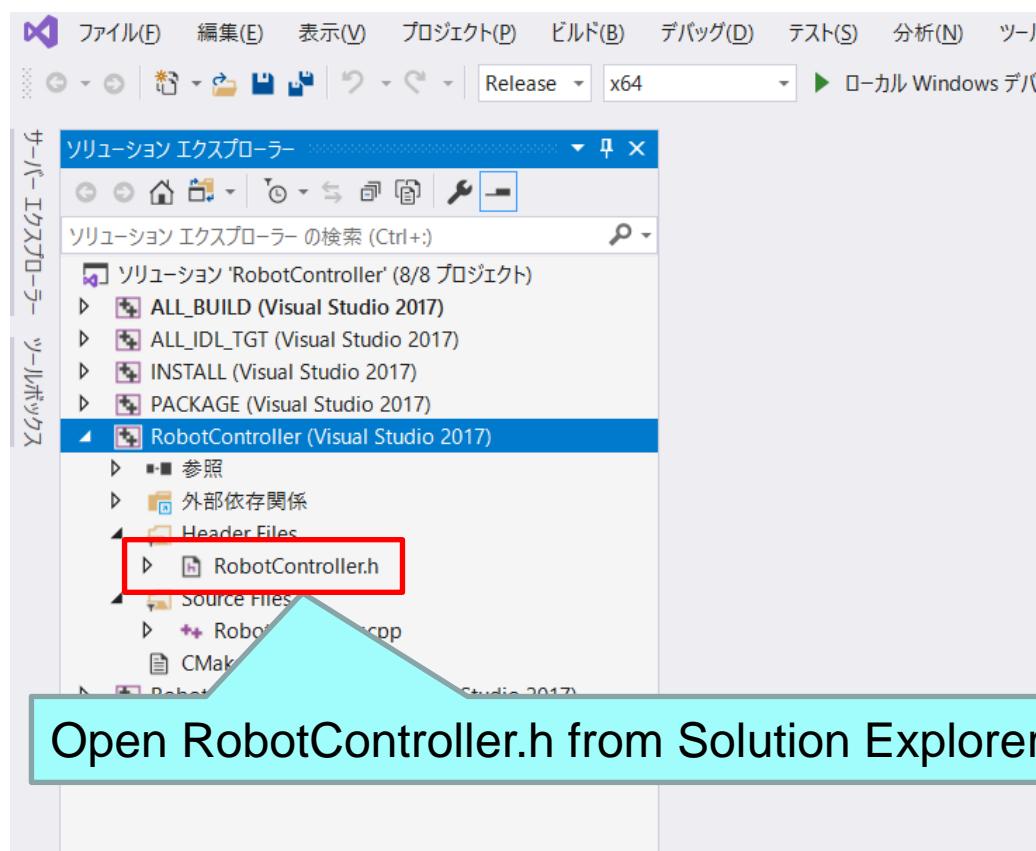
- Windows
  - Visual Studio starts
- Ubuntu
  - Code::Blocks starts



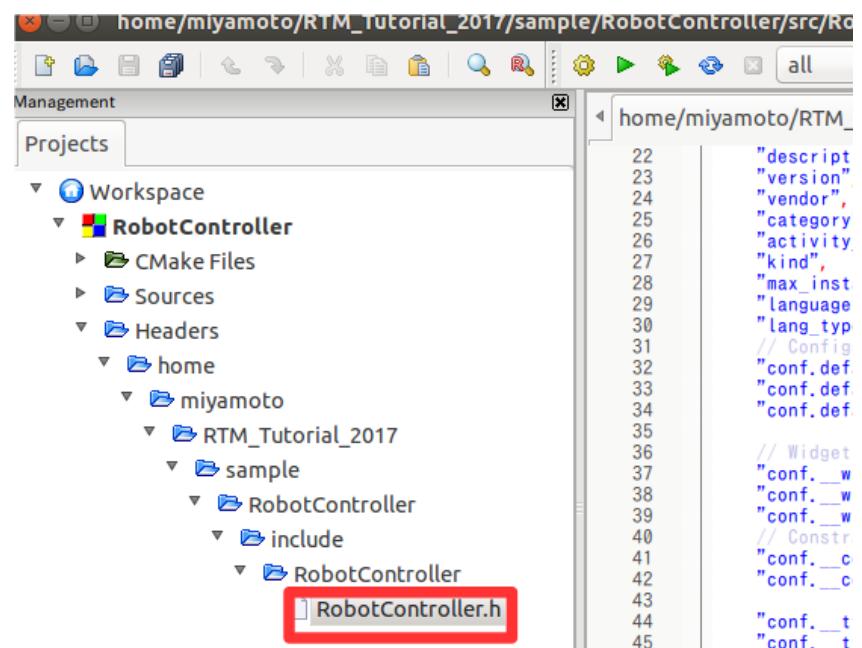
# Editing source code

- Edit RobotController.h

Visual Studio



Code::Blocks



# Editing source code

- Edit RobotController.h

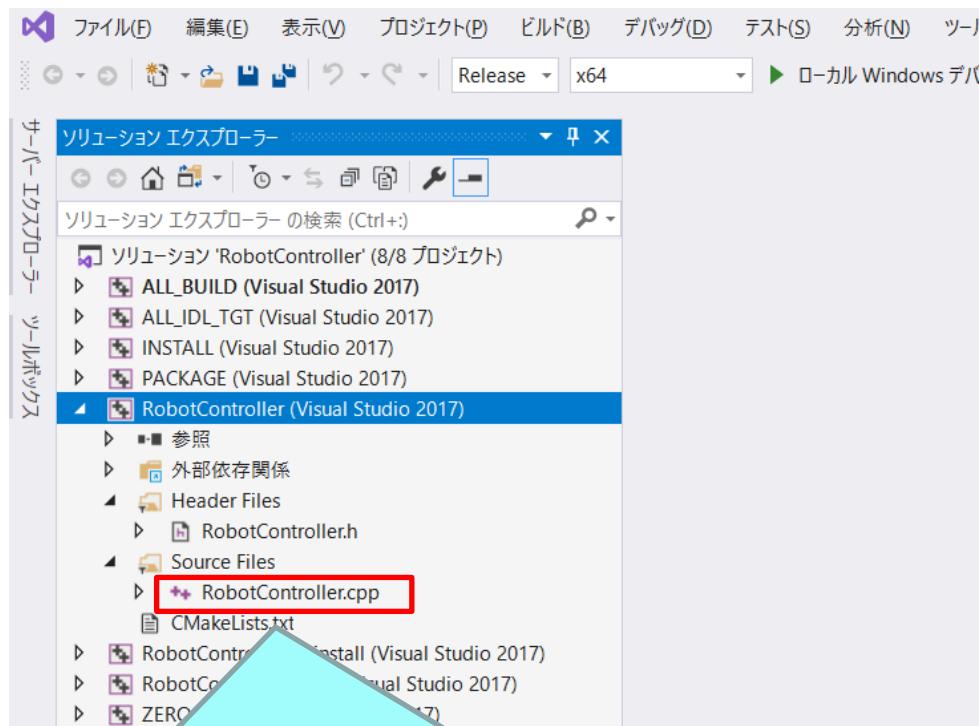
```
265 Declaration of a variable that temporarily stores the sensor value
266
267 int sensor_data[4];
268
```

```
269 ↓
270 private: ↓
271     int sensor_data[4]; ↓
272     // <rtc-template block="private_attribute">↓
273     ↓
274     // </rtc-template>↓
275     ↓
276     // <rtc-template block="private_operation">↓
277     ↓
278     // </rtc-template>↓
279     ↓
280 }; ↓
281 ↓
282 ↓
283 extern "C" ↓
284 { ↓
```

# Editing source code

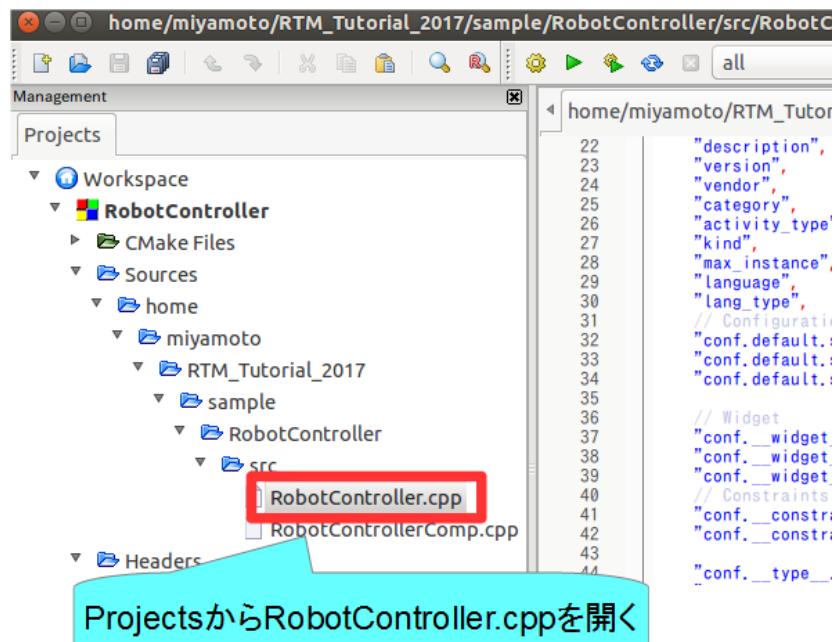
- Edit RobotController.cpp

Visual Studio



Open RobotController.cpp from Solution Explorer

Code::Blocks



ProjectsからRobotController.cppを開く

# Editing source code

- RobotController.cppの編集

```
125 RTC::ReturnCode_t RobotController::onActivated(RTC::UniqueId ec_id) {  
126     //Sensor value initialization  
127     for (int i = 0; i < 4; i++) {  
128         sensor_data[i] = 0;  
129     }  
130     return RTC::RTC_OK;  
131 }  
132  
133 // Added to onActivated function
```

```
137 RTC::ReturnCode_t RobotController::onDeactivated(RTC::UniqueId ec_id) {  
138     //Stop the robot  
139     m_out.data.vx = 0;  
140     m_out.data.va = 0;  
141     m_outOut.write();  
142     return RTC::RTC_OK;  
143 }  
144  
145 // Added to onDeactivated function
```

# Editing source code

- Edit RobotController.cpp
  - If it is difficult to write, please copy and paste from the following page.
  - <https://openrtm.org/openrtm/ja/node/6550#toc20>

```
148 RTC::ReturnCode_t RobotController::onExecute(RTC::UniqueId ec_id) {  
149     //Confirmation of existence of input data  
150     if (m_inIn.isNew()) {  
151         //Input data read  
152         m_inIn.read();  
153         //At this point the input data is stored in m_in.  
154         for (int i = 0; i < m_in.data.length(); i++) {  
155             //Store input data in another variable  
156             if (i < 4) {  
157                 sensor_data[i] = m_in.data[i];  
158             }  
159         }  
160     }  
161     //Determine if you want to stop only when moving forward  
162     if (m_speed_x > 0) {  
163         for (int i = 0; i < 4; i++) {  
164             //Determine if the sensor value is greater than or equal to the set value  
165             if (sensor_data[i] > m_stop_d) {  
166                 //Stop if the sensor value is greater than or equal to the set value  
167                 m_out.data.vx = 0;  
168                 m_out.data.va = 0;  
169                 m_outOut.write();  
170                 return RTC::RTC_OK;  
171             }  
172         }  
173     }  
174     //If there is no sensor with a value higher than the set value, operate with the value of the configuration parameter.  
175     m_out.data.vx = m_speed_x;  
176     m_out.data.va = m_speed_r;  
177     m_outOut.write();  
178     return RTC::RTC_OK;  
179 }  
180  
181  
182  
183  
184  
185  
186  
187 }
```

Added to onExecute function

# Editing source code

- Procedure to read data

```
148 RTC::ReturnCode_t Read()
149 {
150     //Check if there is newly written data with the isNew function
151     if (m_inIn.isNew())
152     {
153         //Confirmation of existence of input data
154         //Input data read
155         m_inIn.read();
156         //At this point the input data is stored in m_in.
157         for (int i = 0; i < m_in.data.length(); i++)
158         {
159             //Data is stored in the variable m_in when the read function is called.
160             sensor_data[i] = m_in.data[i];
161         }
162     }
163 }
164
165 
```

TimedShortSeq type holds multiple data like an array.

# Editing source code

- Procedure for writing data

```
//Determine if you want to write data
if (m_speed_x > 0) {
    for (int i = 0; i < sensor_data.size(); i++) {
        //Determine if the sensor value is greater than or equal to the set value
        if (sensor_data[i] >= m_stop_d) {
            //Stop if the sensor value is greater than or equal to the set value
            m_out.data.vx = 0;
            m_out.data.va = 0;
            m_outOut.write();
            return RTC::RTC_OK;
        }
    }
    //If there is no sensor with a value
    m_out.data.vx = m_speed_x;
    m_out.data.va = m_speed_r;
    m_outOut.write();
    return RTC::RTC_OK;
}
```

Store data in the variable m\_out.

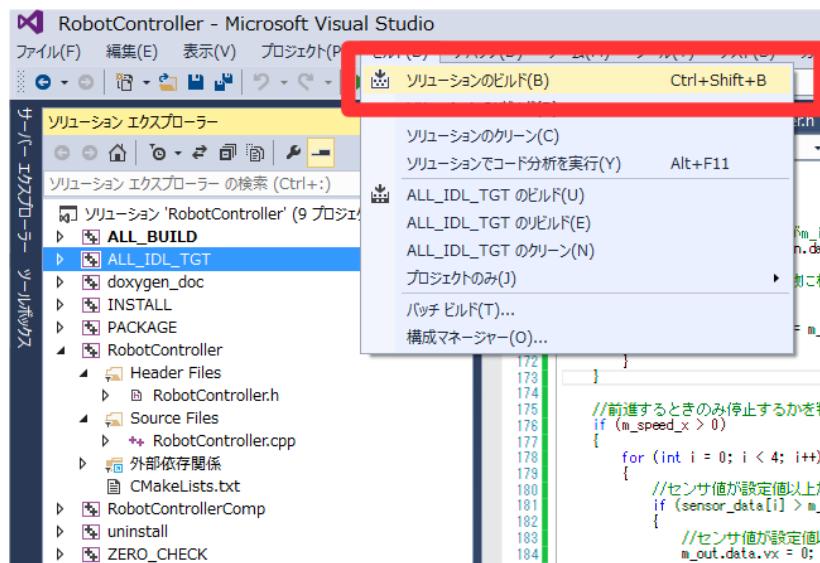
Since it is a TimedVelocity2D type, the straight speed is stored in vx and the rotation speed is stored in va.

When the configuration parameter is changed, the value is stored in the corresponding variable (m\_speed\_x, m\_speed\_r, m\_stop\_d).

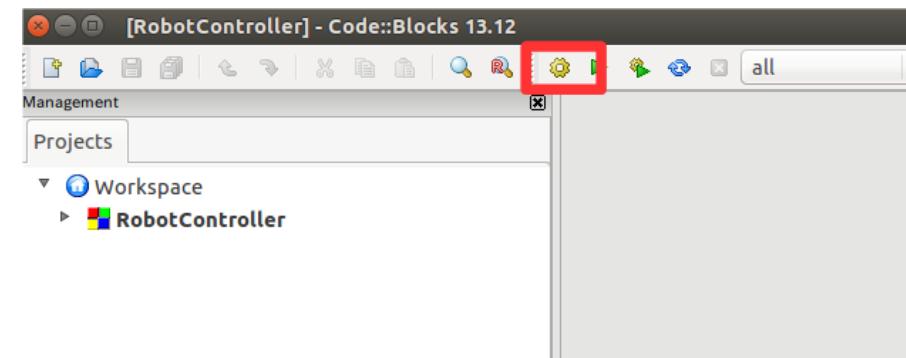
Write data with write function

# RTC build

## Visual Studio



## Code::Blocks



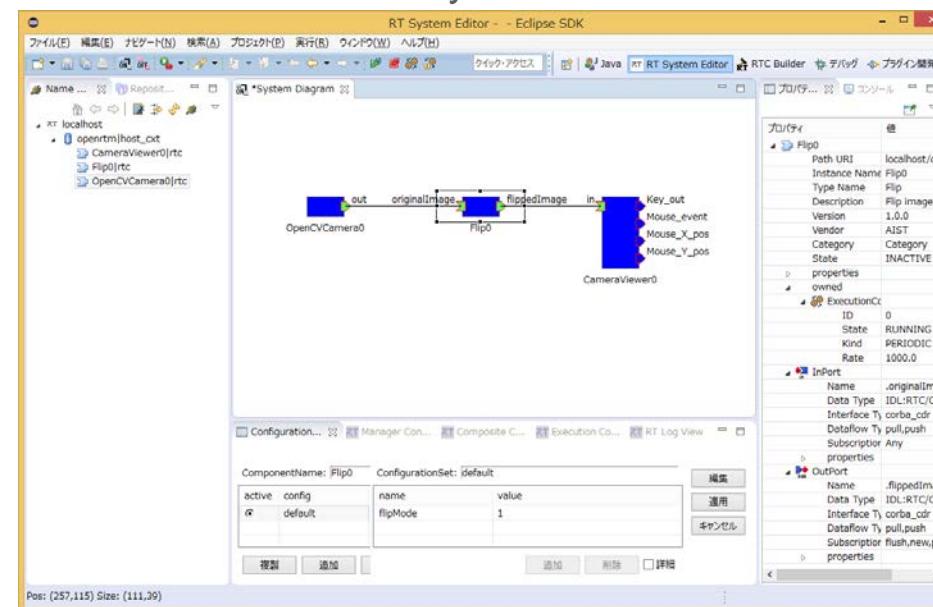
- If successful, an executable file will be generated.
  - Windows
    - RobotControllerComp.exe is generated in the Release (or Debug) folder of the build\src folder.
  - Ubuntu
    - RobotControllerComp is generated in the build/src folder

# System construction support tool

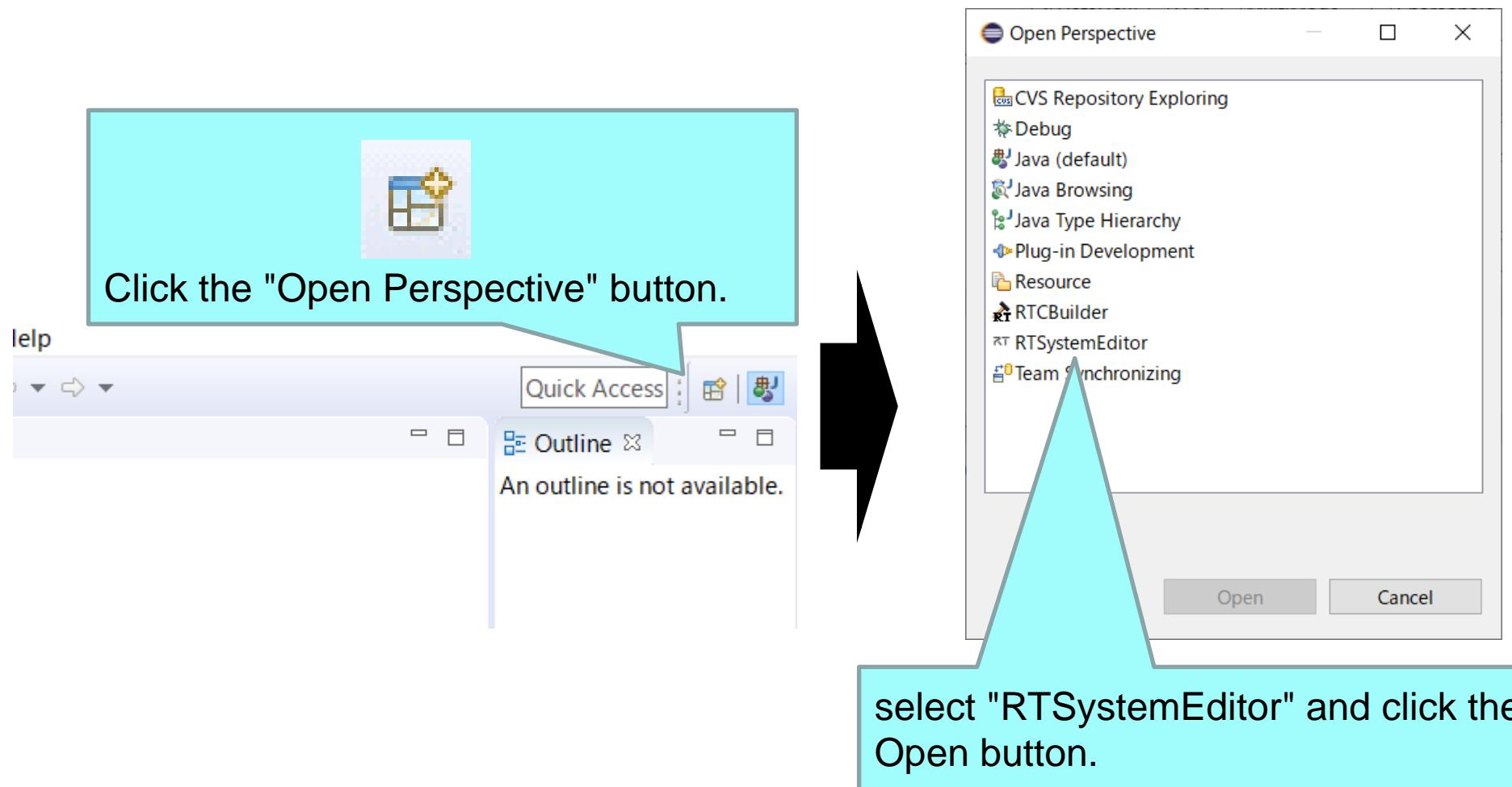
## RT System Editor

# 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

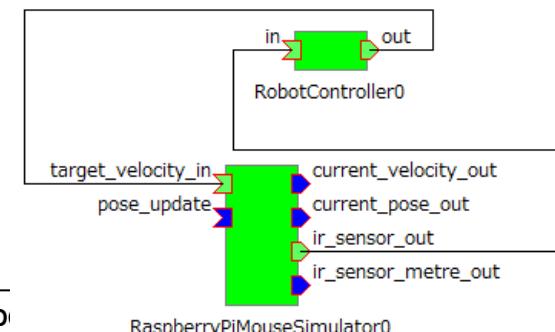


# RT System Editor screen configuration



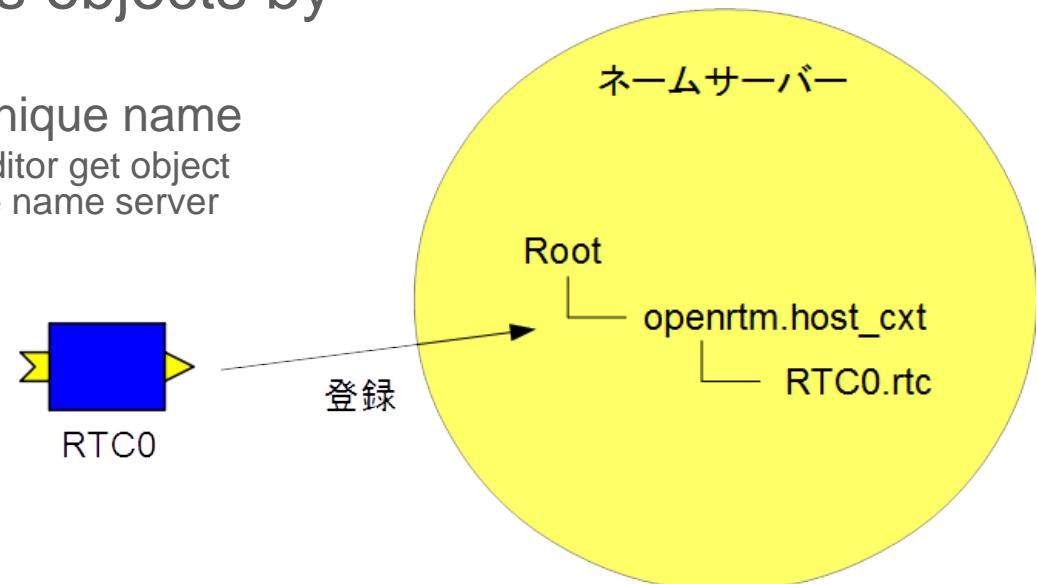
# Operation check of RobotController component

- Create an RT system that connects to simulator components and operates robots on the simulator
  - Start Naming Service
  - Start RaspberryPiMouseSimulator Component
    - Windows
      - In the EXE folder of the extracted ZIP file
      - Double-click "RaspberryPiMouseSimulatorComp.exe"
    - Ubuntu
      - If not installed
        - » \$ wget [https://raw.githubusercontent.com/OpenRTM/RTM\\_Tutorial\\_ROBOME\\_CH2019/master/script/install\\_raspimouse\\_simulator.sh](https://raw.githubusercontent.com/OpenRTM/RTM_Tutorial_ROBOME_CH2019/master/script/install_raspimouse_simulator.sh)
        - » \$ sh install\_raspimouse\_simulator.sh
      - Go to the RasPiMouseSimulatorRTC directory and enter the following command
        - » \$ build/src/RaspberryPiMouseSimulatorComp
    - Start RobotController Component
    - Connect the RaspberryPiMouseSimulator component and RobotController component and execute "All Activate"

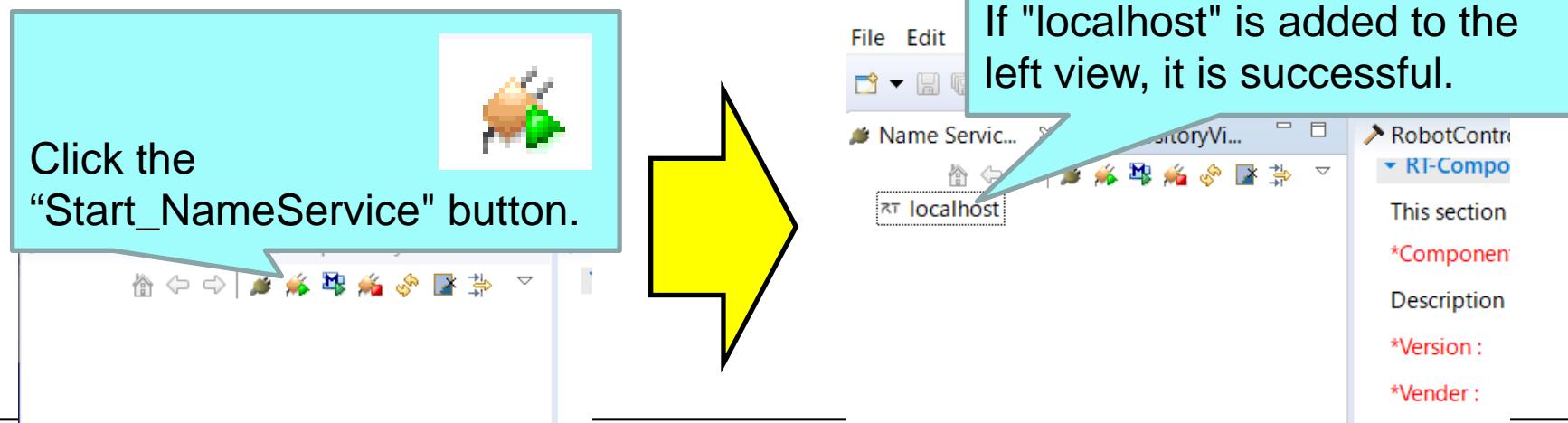


# Naming Service

- 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

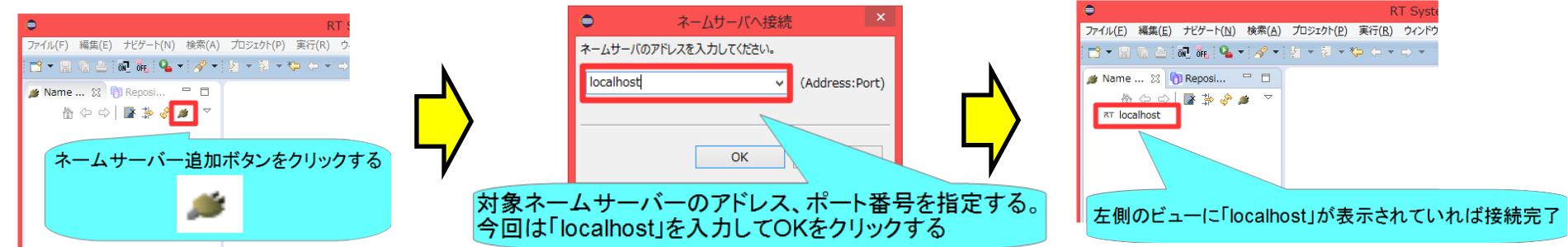


- Procedure to start



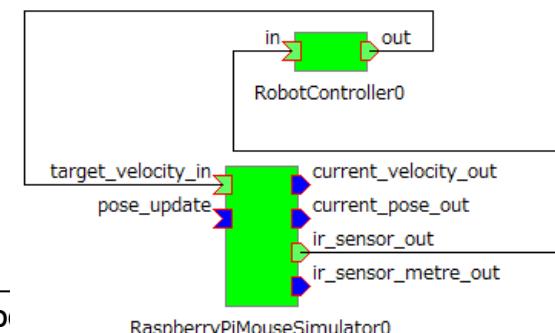
# Start Naming Service

- 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



# Operation check of RobotController component

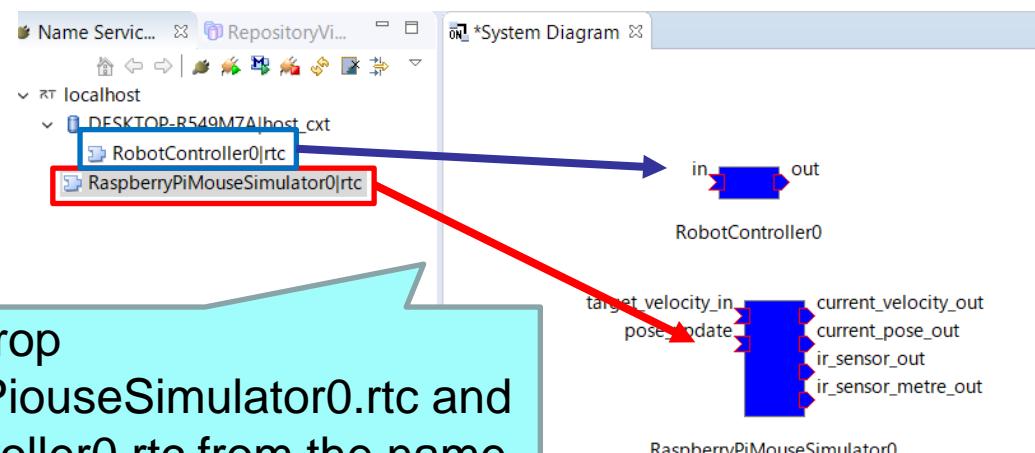
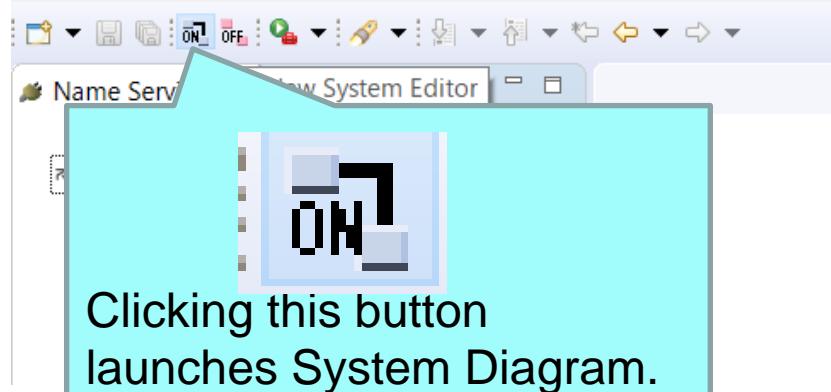
- Create an RT system that connects to simulator components and operates robots on the simulator
  - Start Naming Service
  - Start RaspberryPiMouseSimulator Component
    - Windows
      - In the EXE folder of the extracted ZIP file
      - Double-click "RaspberryPiMouseSimulatorComp.exe"
    - Ubuntu
      - If not installed
        - » \$ wget [https://raw.githubusercontent.com/OpenRTM/RTM\\_Tutorial\\_ROBOME\\_CH2019/master/script/install\\_raspimouse\\_simulator.sh](https://raw.githubusercontent.com/OpenRTM/RTM_Tutorial_ROBOME_CH2019/master/script/install_raspimouse_simulator.sh)
        - » \$ sh install\_raspimouse\_simulator.sh
      - Go to the RasPiMouseSimulatorRTC directory and enter the following command
        - » \$ build/src/RaspberryPiMouseSimulatorComp
    - Start RobotController Component
    - Connect the RaspberryPiMouseSimulator component and RobotController component and execute "All Activate"



# Data port connection

workspace6 - Eclipse SDK

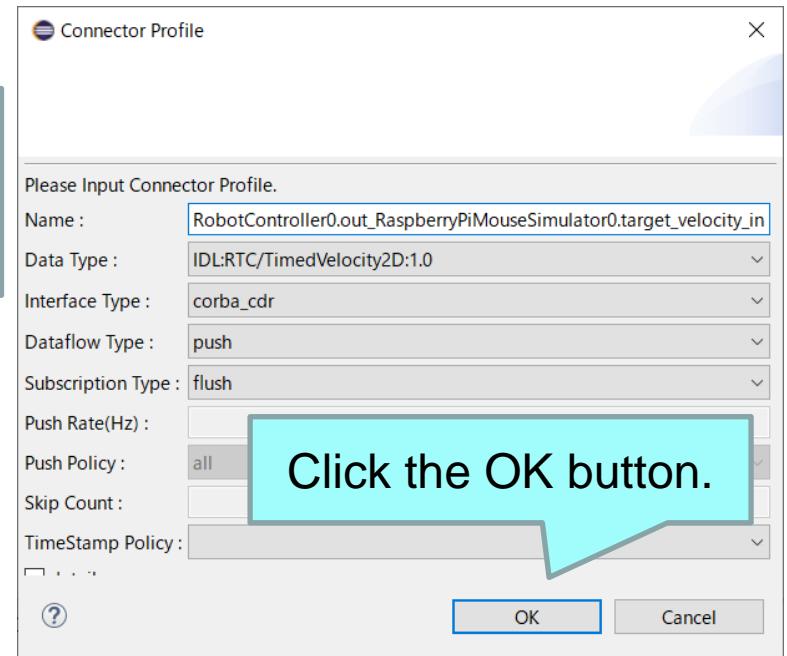
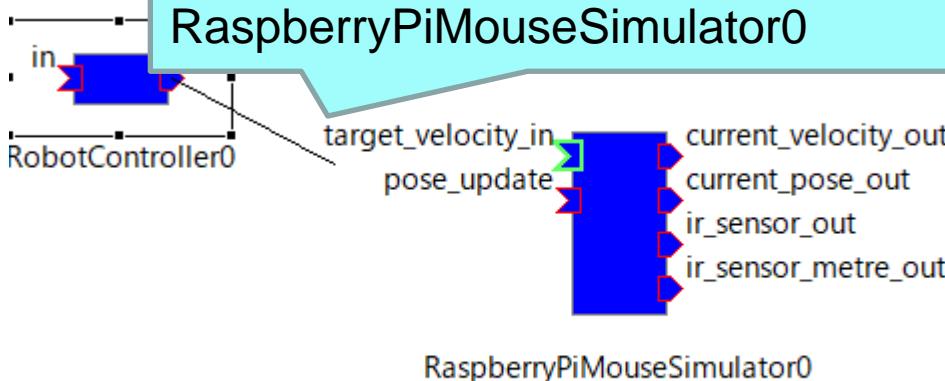
File Edit Navigate Search Project Run Window Help



Drag and drop  
RaspberryPiouseSimulator0.rtc and  
RobotController0.rtc from the name  
service view on the left to the  
System Diagram.

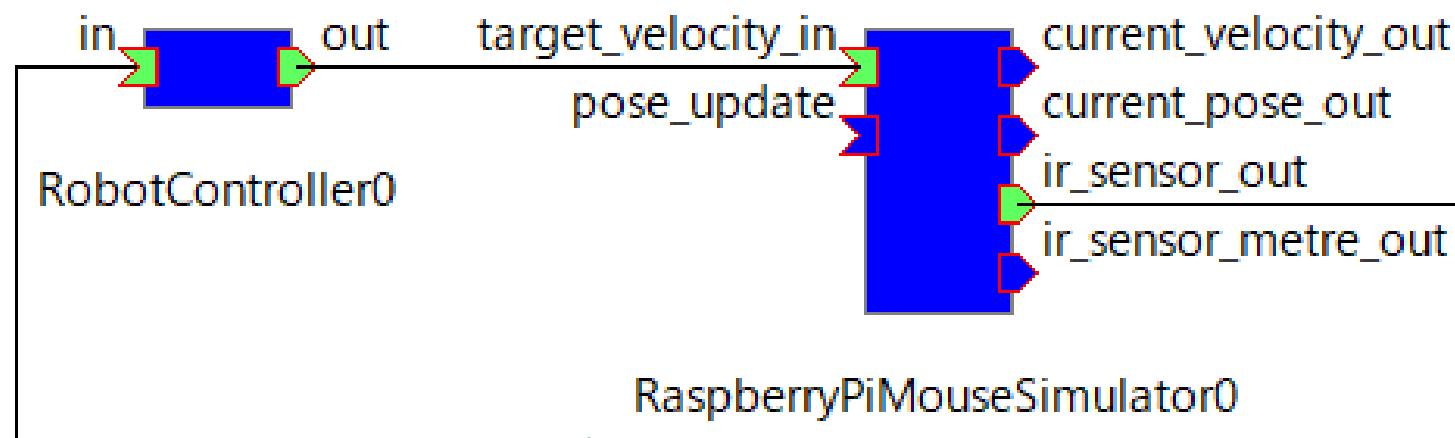
# Data port connection

Drag and drop "out" of RobotController0 to "target\_velocity\_in" of RaspberryPiMouseSimulator0



1. A line appears between the ports
2. InPort and OutPort are displayed in green

# Data port connectionの接続

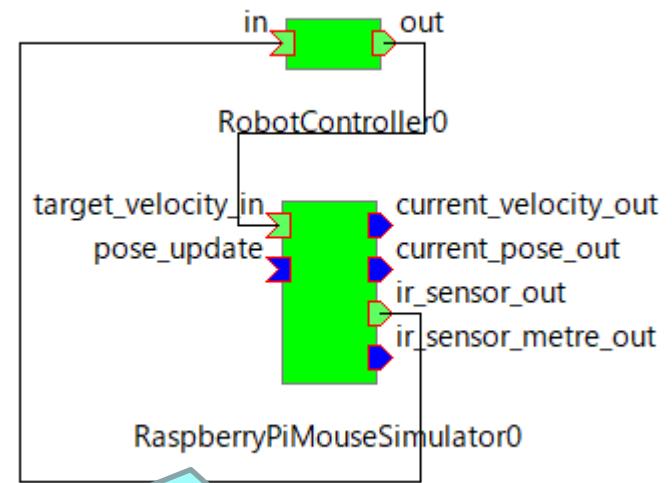
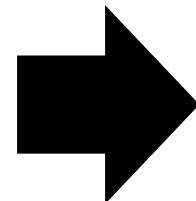
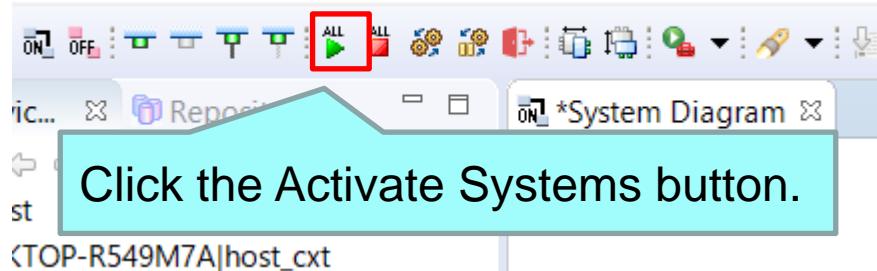


Connect "ir\_sensor\_out" of RaspberryPiSimulator0 and "in" of RobotController0.

# Activation

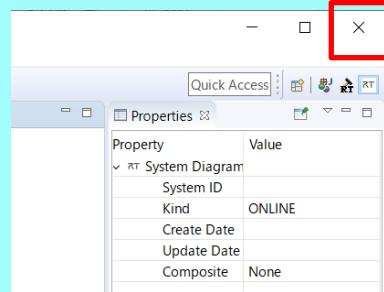
e6 - - Eclipse SDK

Navigate Search Project Run Window Help

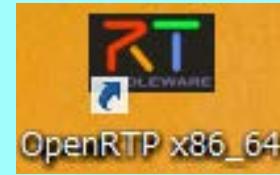


If the RTC changes green, activation is successful.

If the button does not appear, restart OpenRTP.



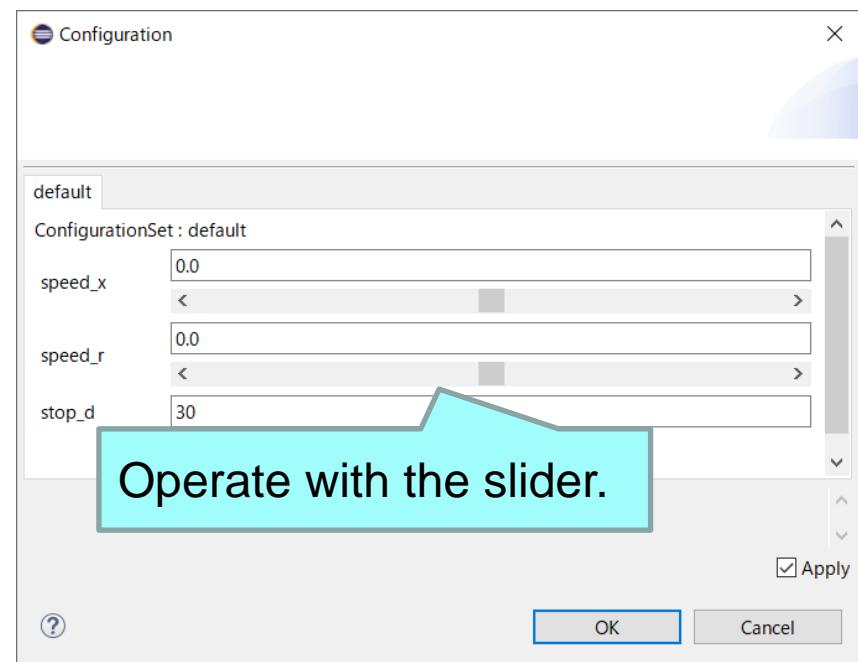
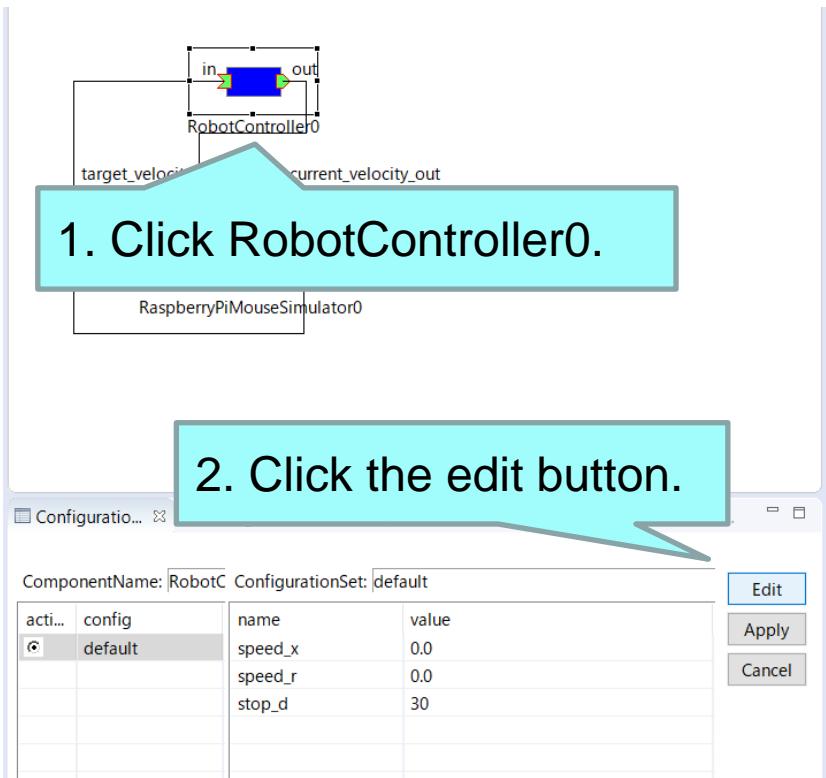
Exit OpenRTP.



Double-click on the shortcut.

# Change configuration parameters

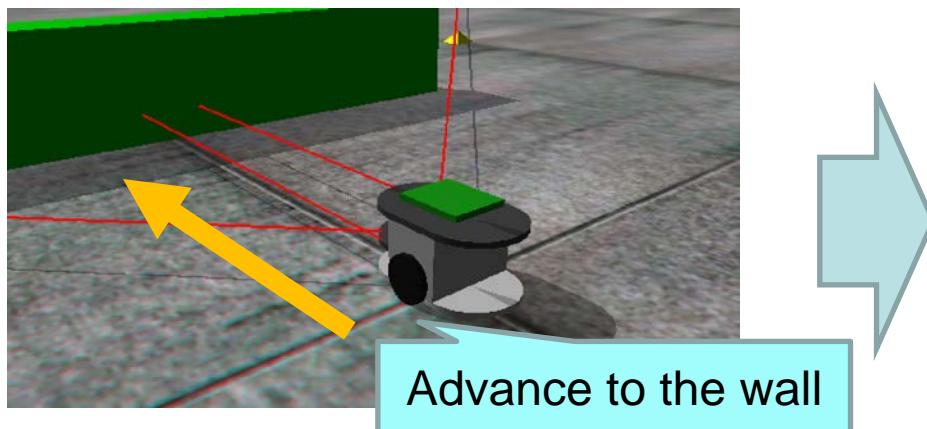
- Change configuration parameters using RT System Editor



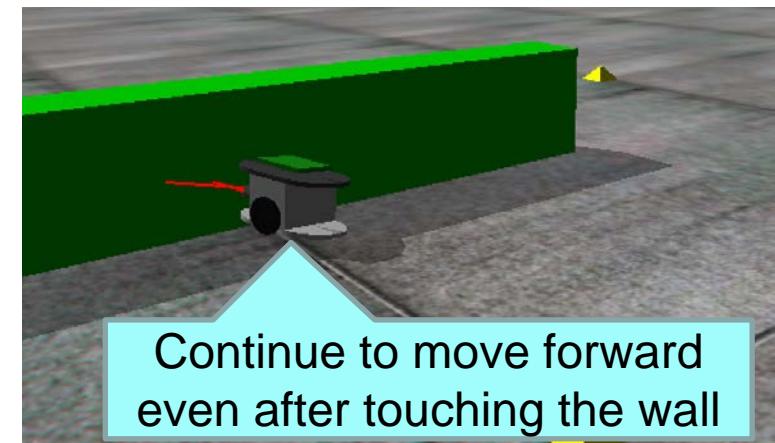
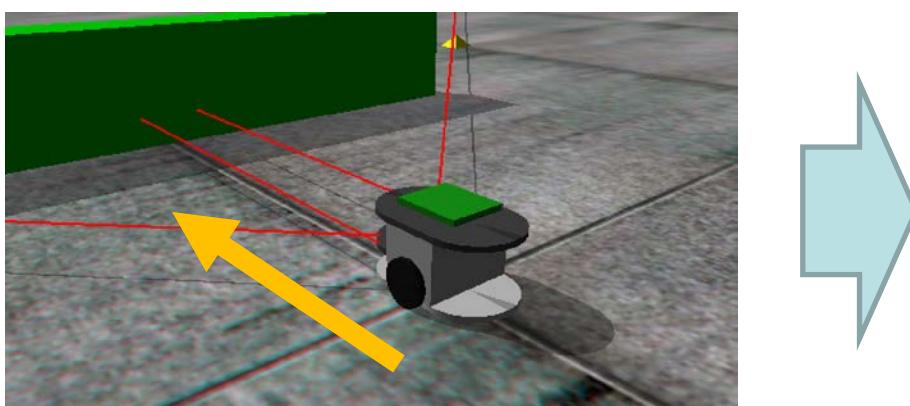
- Check if the following operations can be performed
  - Can the robot on the simulator be operated with the slider?
  - Does the robot stop when it approaches an wall?

# 動作確認

- When it stops when the wall approaches the distance sensor

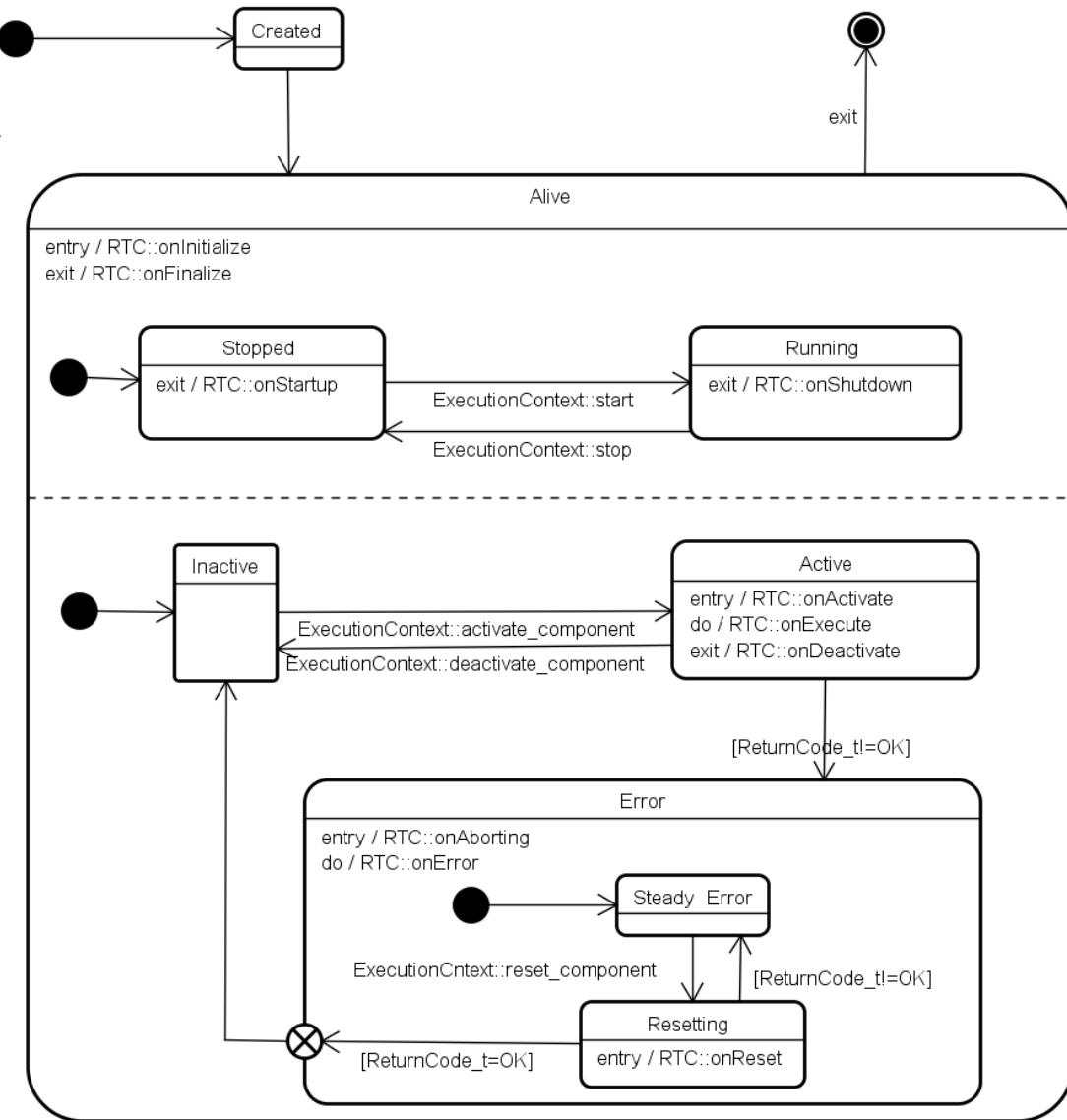


- When the wall does not stop when approaching the distance sensor

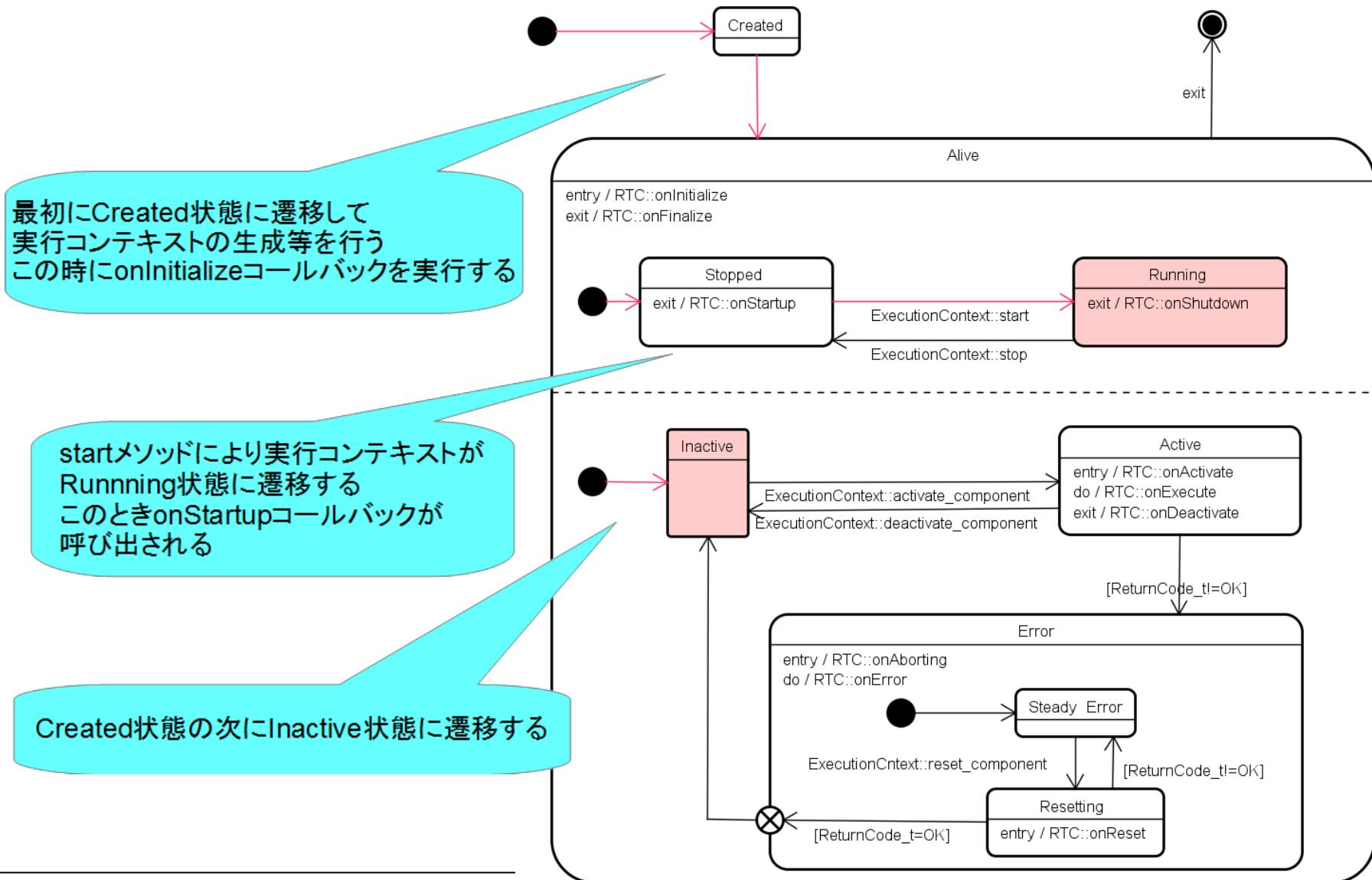


# RTコンポーネントの状態遷移

- RTCには以下の状態が存在する
  - Created**
    - 生成状態
    - 実行コンテキストを生成し、start()が呼ばれて実行コンテキストのスレッドが実行中(Running)状態になる
    - 自動的にInactive状態に遷移する
  - Inactive**
    - 非活性状態
    - activate\_componentメソッドを呼び出すと活性状態に遷移する
    - RT System Editor上での表示は青
  - Active**
    - 活性状態
    - onExecuteコールバックが実行コンテキストにより実行される
    - リターンコードがRTC\_OK以外の場合はエラー状態に遷移する
    - RT System Editor上での表示は緑
  - Error**
    - エラー状態
    - onErrorコールバックが実行コンテキストにより実行される
    - reset\_componentメソッドを呼び出すと非活性状態に遷移する
    - RT System Editor上での表示は赤
  - 終了状態**



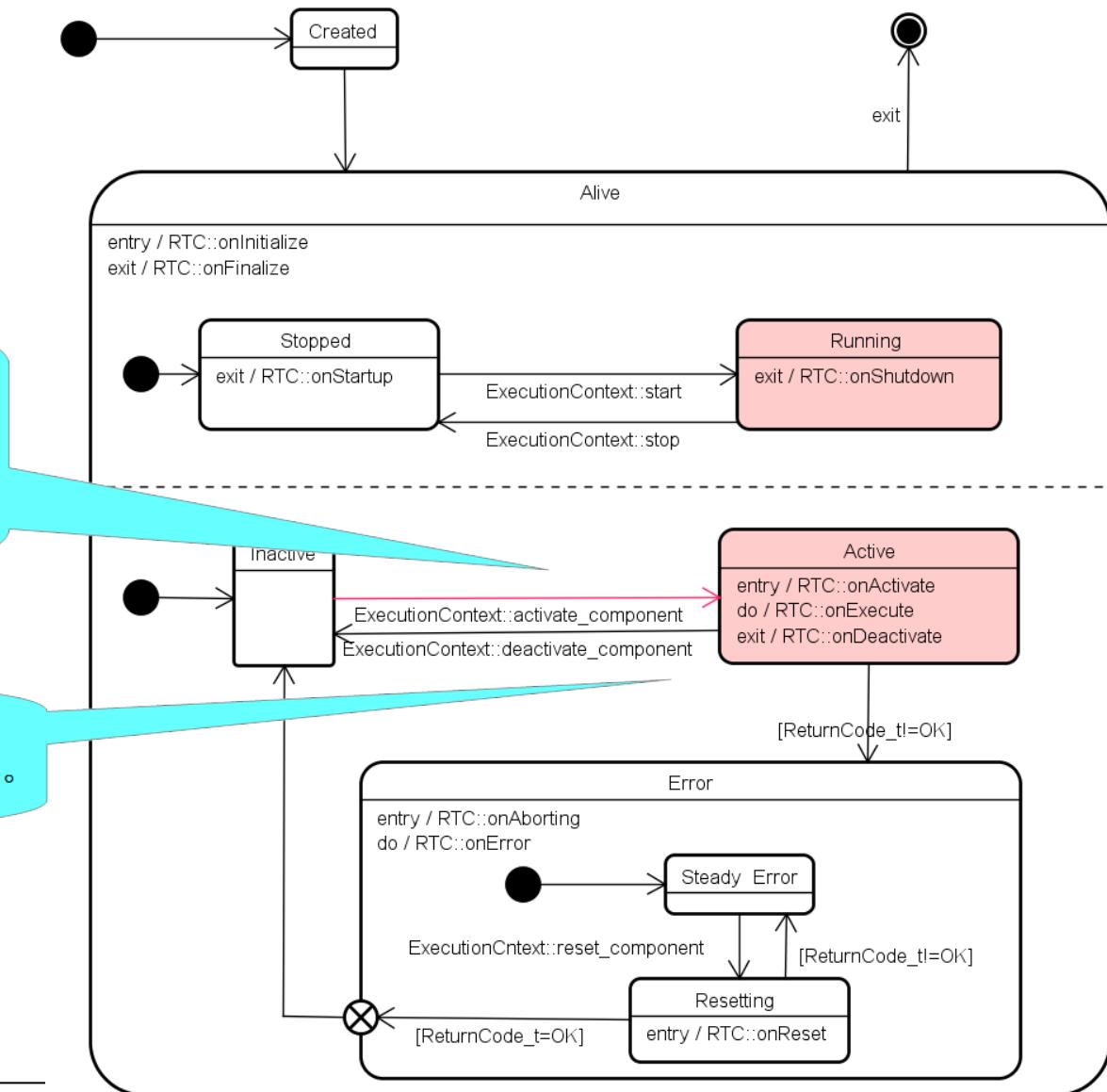
# RTコンポーネントの状態遷移(生成直後)



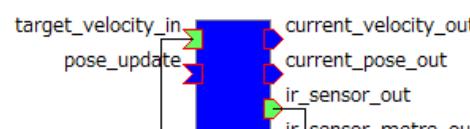
# RTコンポーネントの状態遷移(アクティブ化)

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

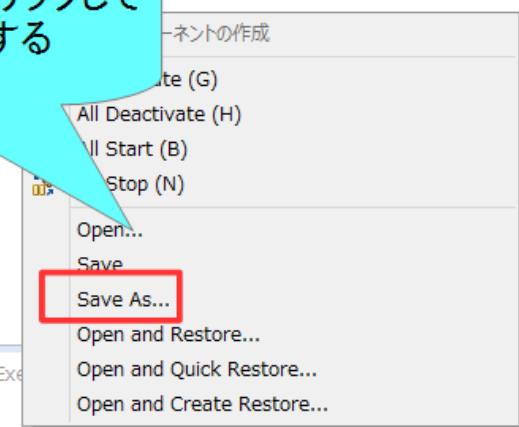
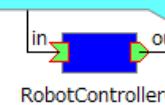
周期実行の実行コンテキストの場合、  
 onExecuteコールバックが周期的に呼び出される。



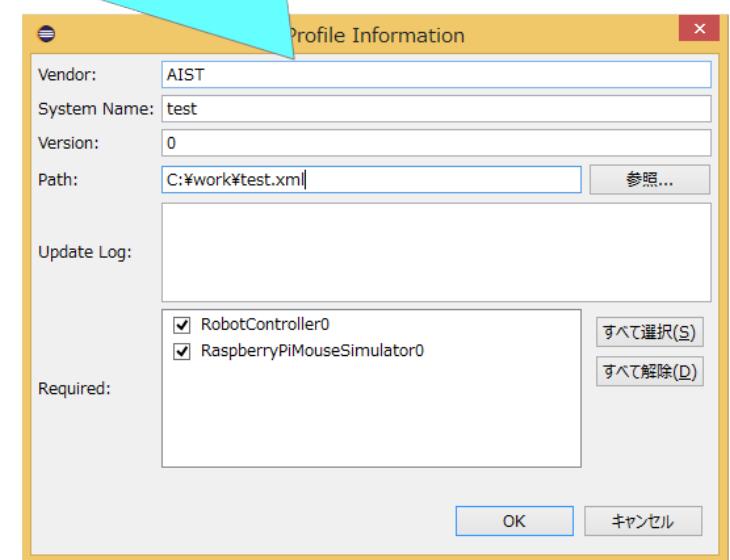
# システムの保存



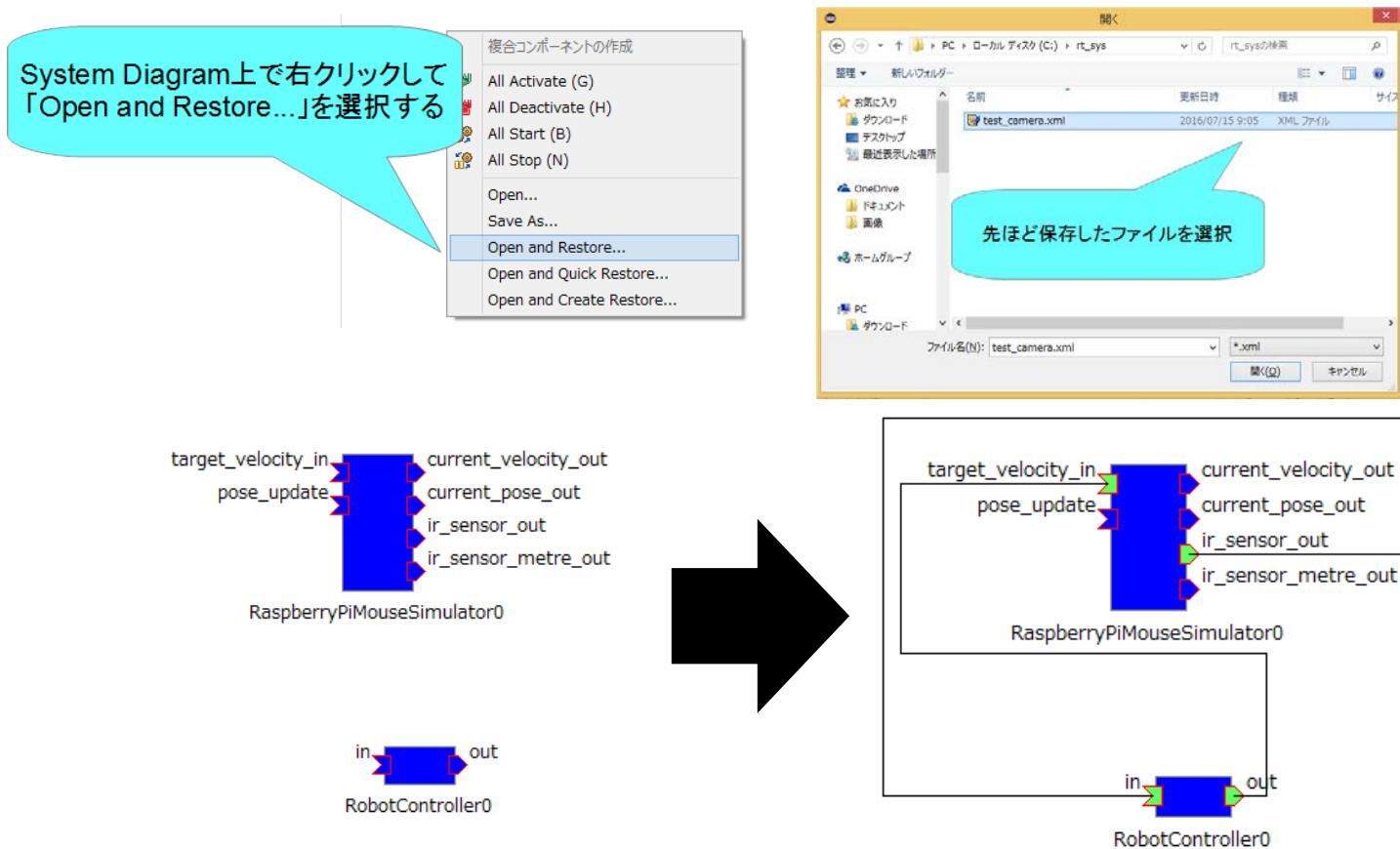
System Diagram上で右クリックして  
「Save As...」を選択する



ベンダ名、システム名、バージョン、保存ファイル名を入力



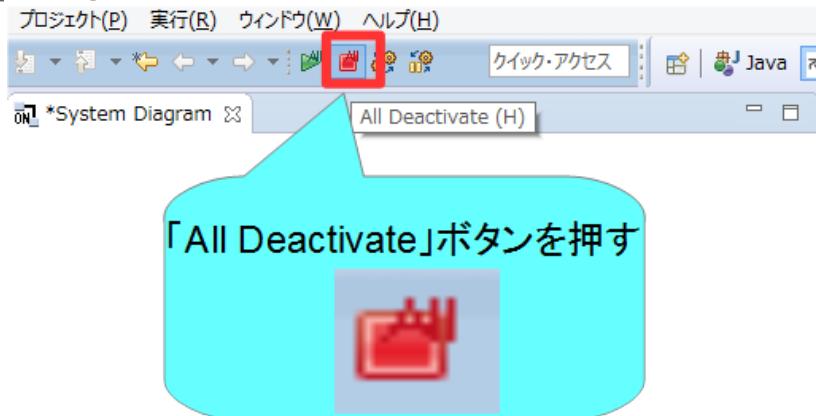
# システムの復元



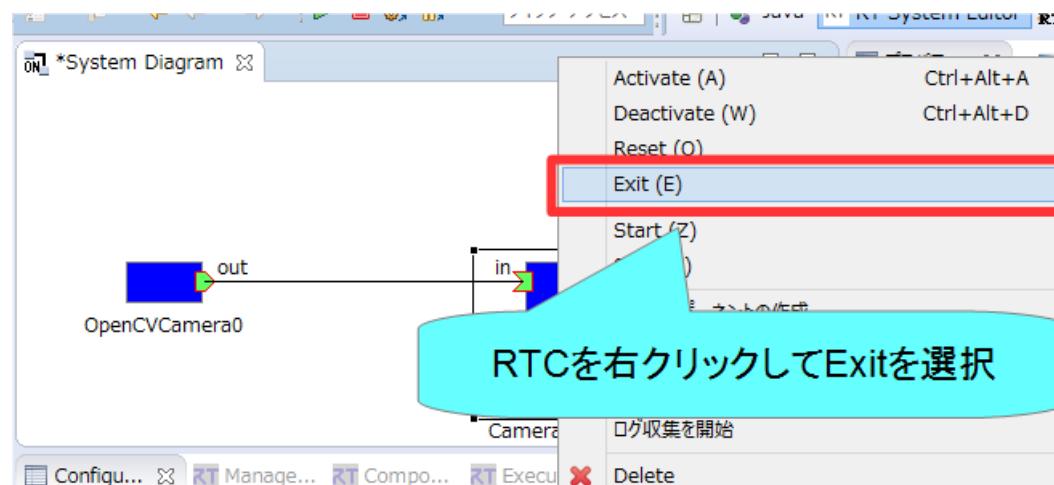
- 以下の内容を復元
  - ポート間の接続
  - コンフィギュレーション
  - 「Open and Create Restore」を選択した場合はマネージャからコンポーネント起動

# 非アクティブ化、終了

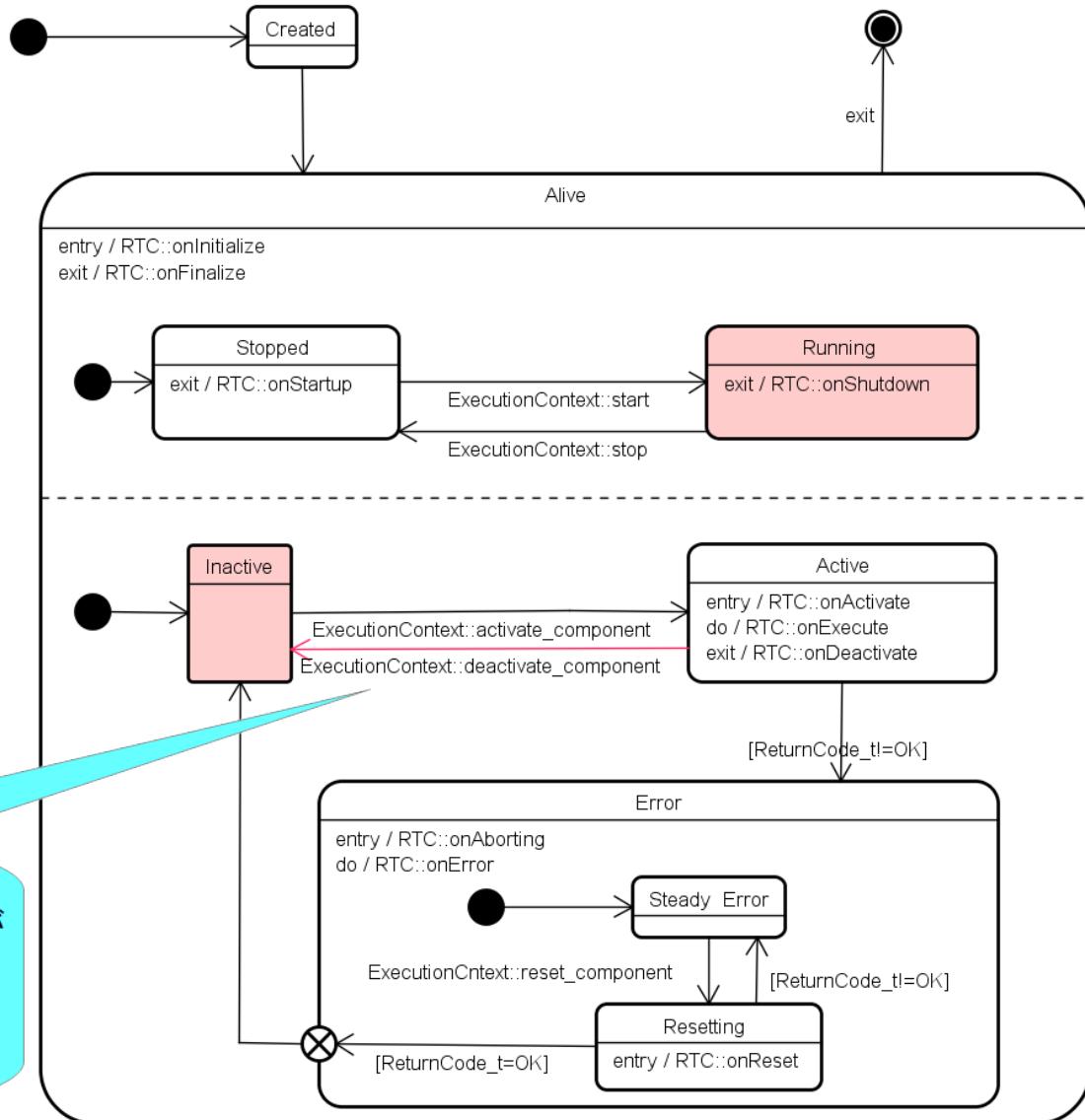
- 非アクティブ化



- 終了



# RTコンポーネントの状態遷移(非アクティブ化)



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

# Raspberry Piマウス実機との接続

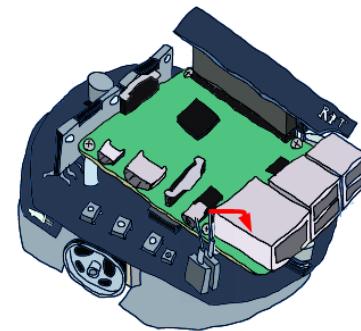
- Raspberry PiとノートPCを無線LANで接続
  - Raspberry Piが無線LANアクセスポイントになる



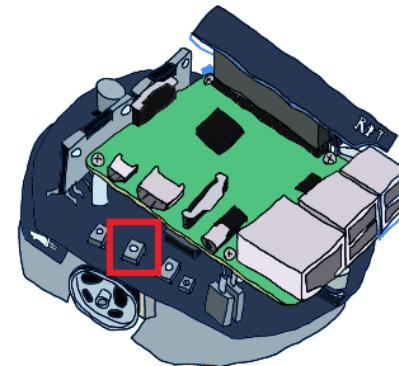
- 注意事項
  - ノートPCに複数のネットワークインターフェースが存在する場合にRTCの通信ができない可能性があります。
    - 問題が発生した場合は個別に対応します。
  - Raspberry Piアクセスポイント接続後はインターネットに接続できなくなります。
  - Raspberry Piアクセスポイント接続後に、**起動済みのネームサーバーとRTCは再起動してください。**
  - Raspberry Piはシャットダウンしてから電源スイッチをオフにするようにしてください
  - モーター電源スイッチはこまめに切るようにしてください

# Raspberry Piとの接続

- Raspberry Piの電源投入
  - 内側のスイッチをオンにする

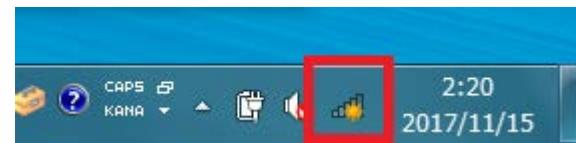


- 電源を切る場合
  - 3つ並んだスイッチの中央のボタンを1秒以上押す
  - 10秒ほどでシャットダウンするため、その後に電源スイッチをオフにする



# Raspberry Piとの接続

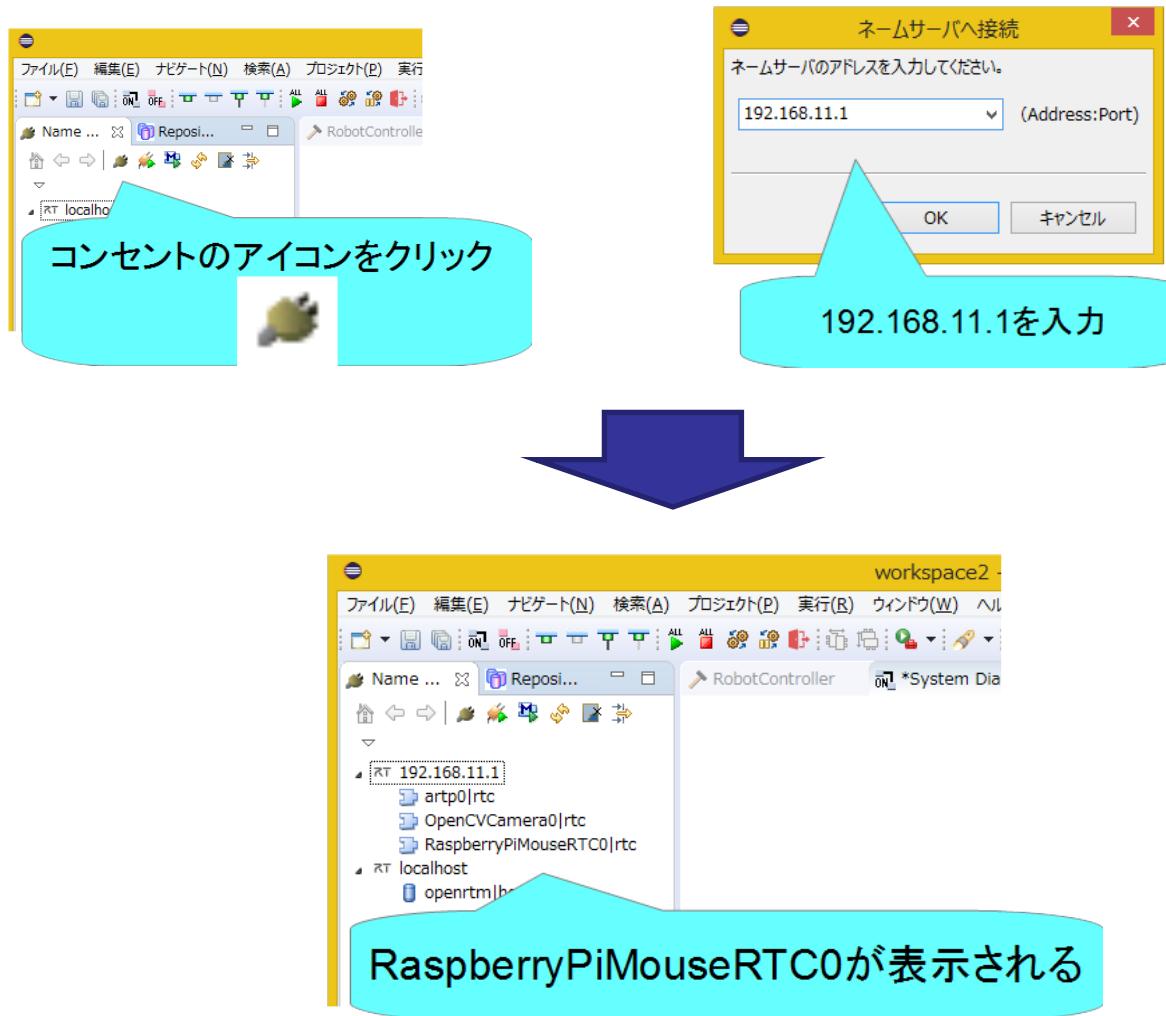
- 無線LANアクセスポイントとの接続
  - SSID、パスワードはRaspberry Piマウス上のシールに記載
  - 接続手順(Windows)
    - 画面右下のネットワークアイコンをクリック



- raspberrypi\_xxに接続後、パスワードを入力

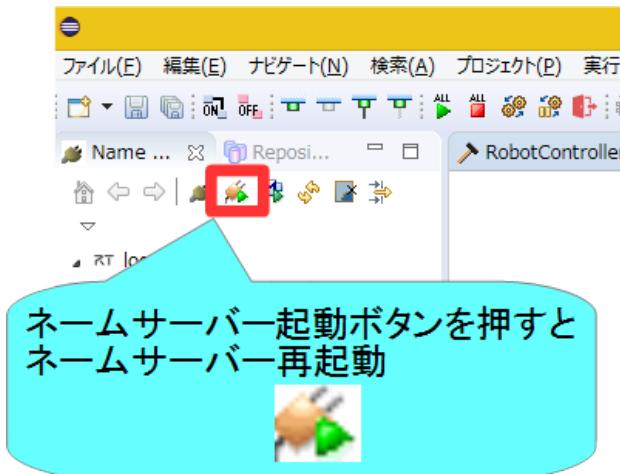


# ネームサーバーとの接続



# 起動済みのRTC、ネームサーバー再起動

- ネームサーバーを再起動する
  - OpenRTM-aist 1.2の場合はネームサーバー起動ボタンで再起動

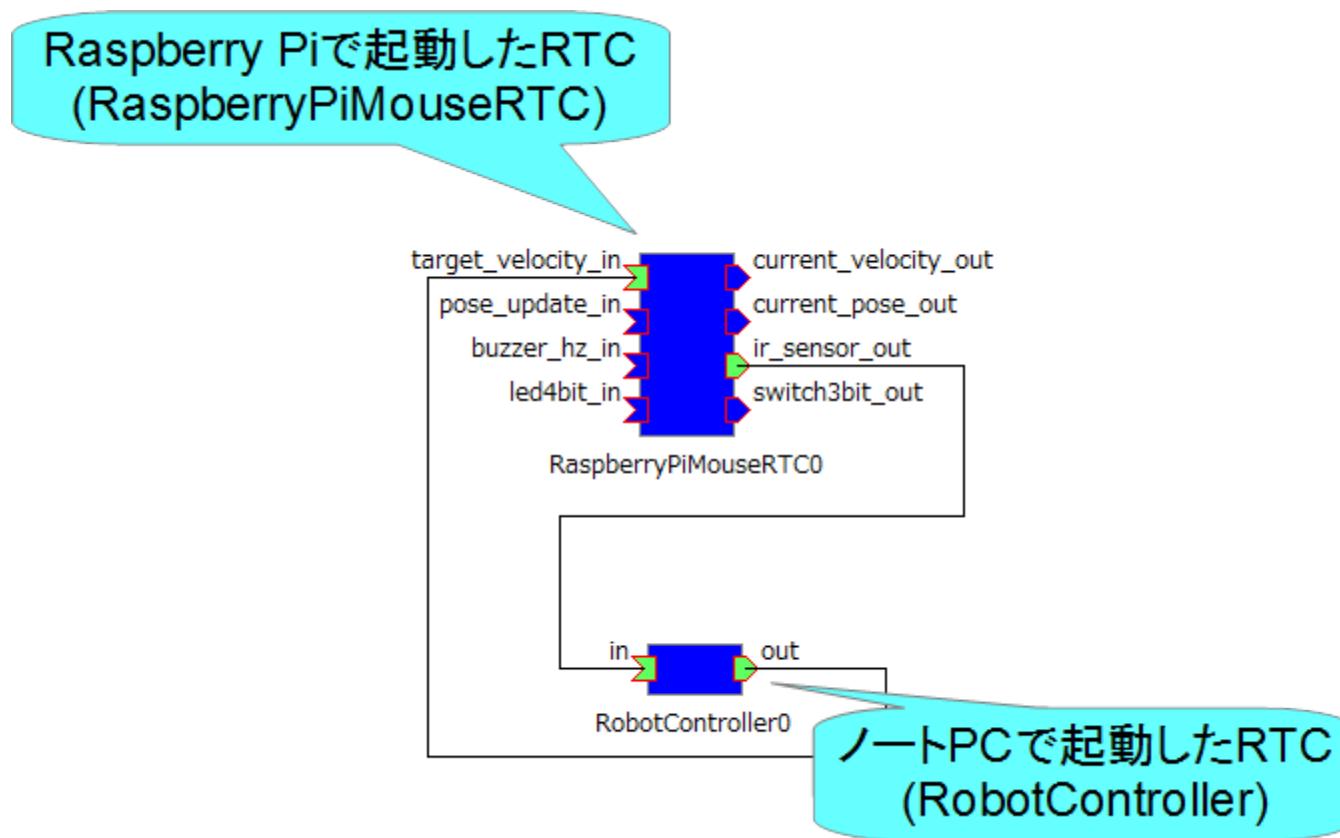


- OpenRTM-aist 1.1.2の場合はネームサーバーのプロセス終了後、「Start Naming Service」を再度実行
- RTC再起動
  - RTCをexitするか、RTC起動時に表示したウィンドウの×ボタンを押して終了する
  - 実行ファイル(RobotControllerComp.exe)を再度実行

```
C:\Users\信澤\Desktop\workspace\SpeechSample\build\src\Release... -> [x]
[27.0.0.1] is the only address available for this server.
: sync_transition: YES
: sync_timeout: 0.5
: type: PeriodicExecutionContext
: rate: 1000
: name:
: port
: port_type: DataOutPort
: dataport
- data_type: IDL:RTC:/timedString:1,0
- subscription_type: fluent,new,periodic
- dataflow_type: push,pull
- interface_type: corba_cdr,direct,shared_memory
```

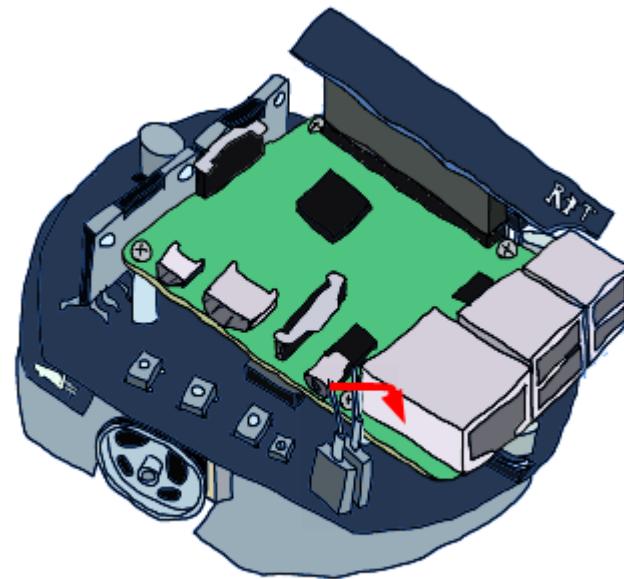
# ポートの接続

- RobotController0とRaspberryPiMouseRTC0を接続する



# 動作確認

- モーターの電源投入
  - 外側のスイッチをONにする



- RTCをアクティブ化して動作確認

# リセット

- RTCがエラー状態に遷移した場合にエディタ上には赤く表示される。



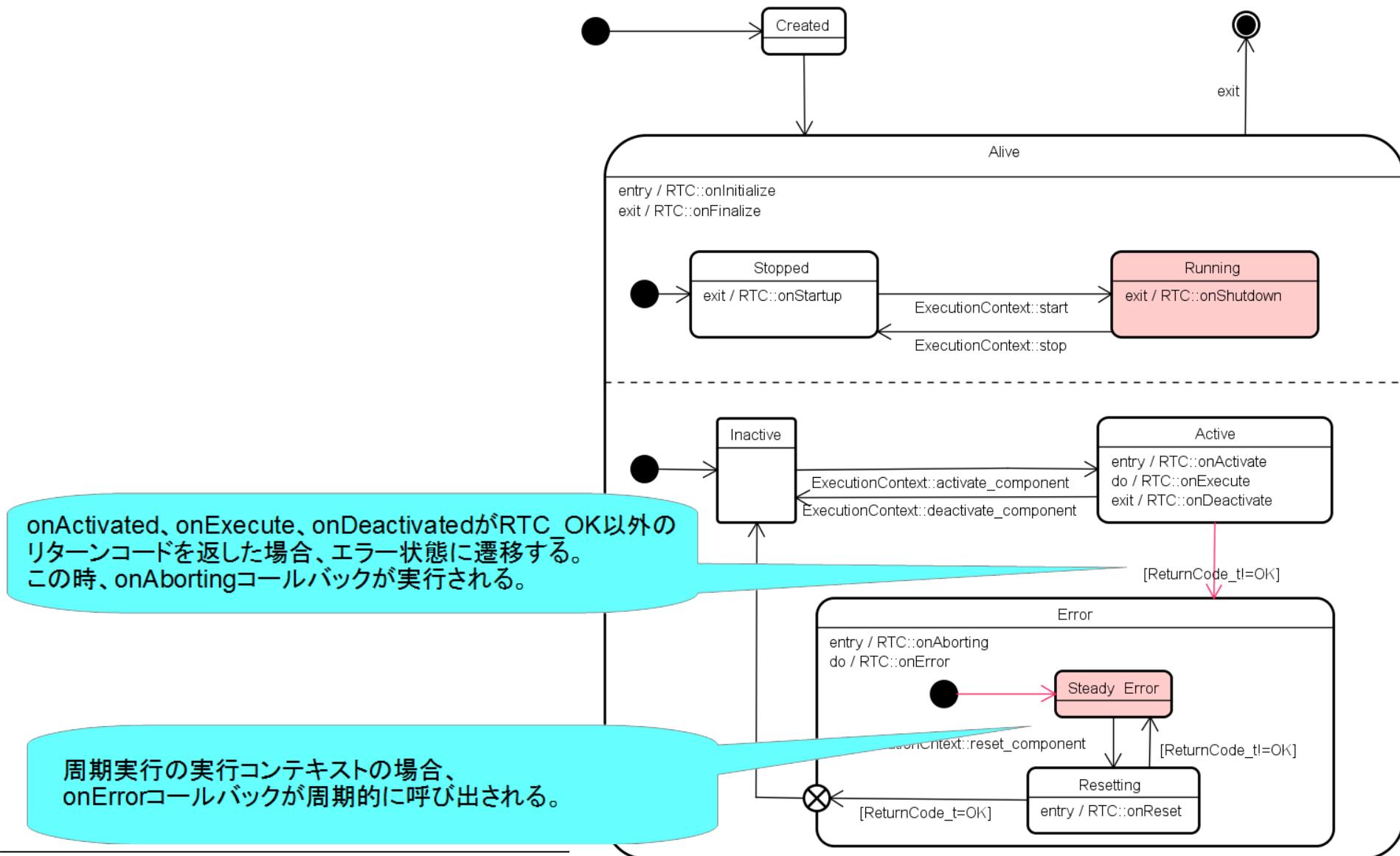
```
RTC::ReturnCode_t Test::onActivated(RTC::UniqueId ec_id) {  
    HANDLE hCom = INVALID_HANDLE_VALUE;  
    hCom = CreateFile("COM5", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);  
    if (hCom == INVALID_HANDLE_VALUE)  
    {  
        return RTC::RTC_ERROR;  
    }
```

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

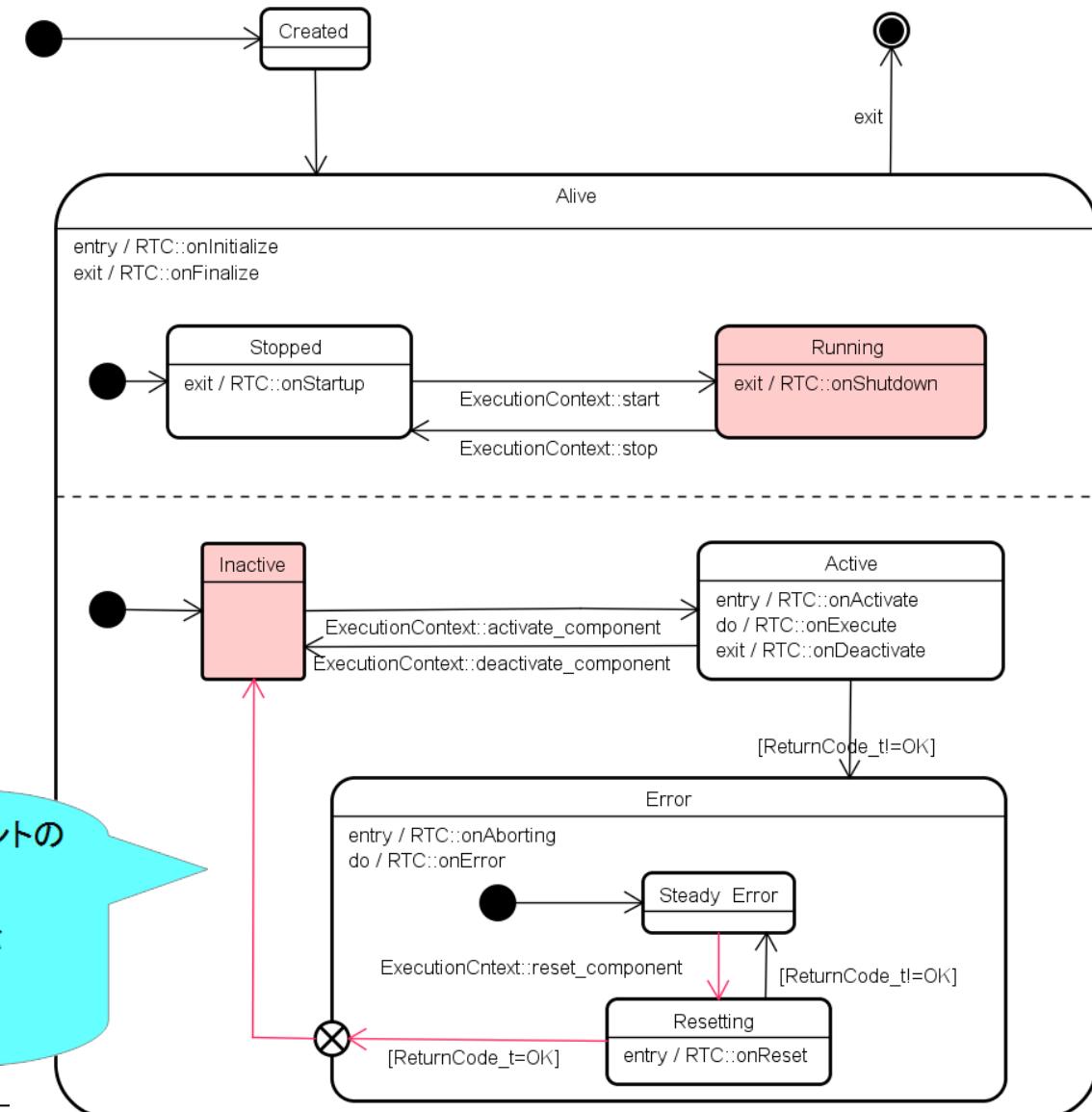
- 以下の操作で非アクティブ状態に戻す



# RTコンポーネントの状態遷移(エラー)



# RTコンポーネントの状態遷移(リセット)



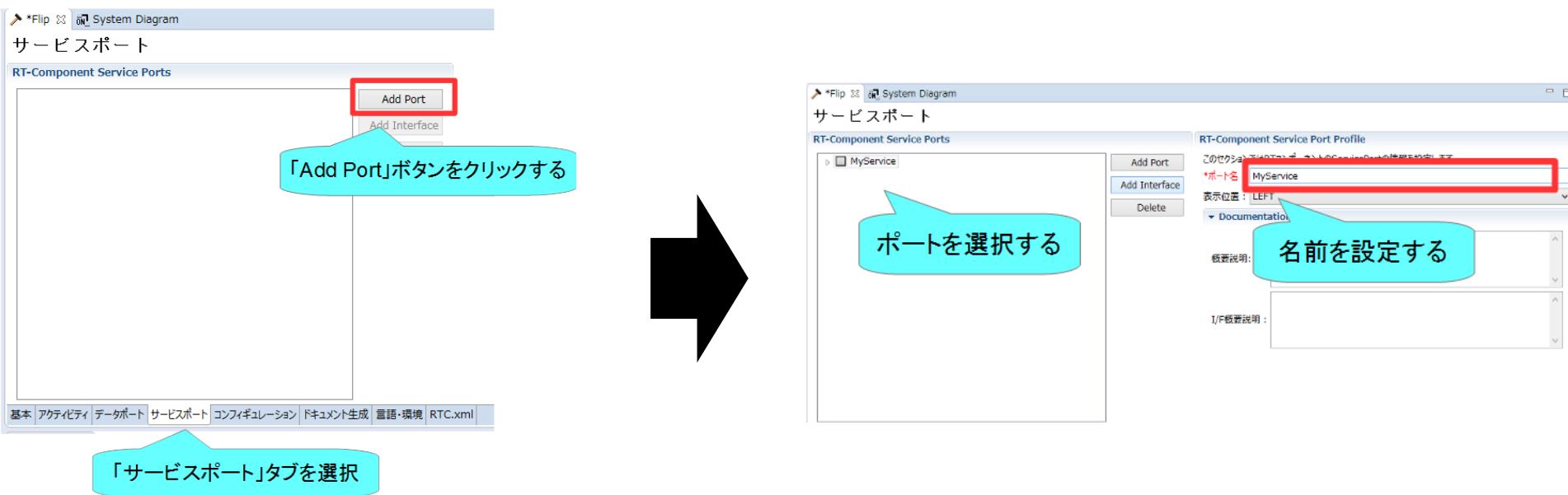
RTシステムエディタの操作によりRTコンポーネントのリセットを行うと`reset_component`メソッドが呼び出される。  
`reset_component`メソッドによりコンポーネントが`Inactive`状態に遷移する。  
この時`onReset`コールバックが実行される

# RTC Builder

## 補足

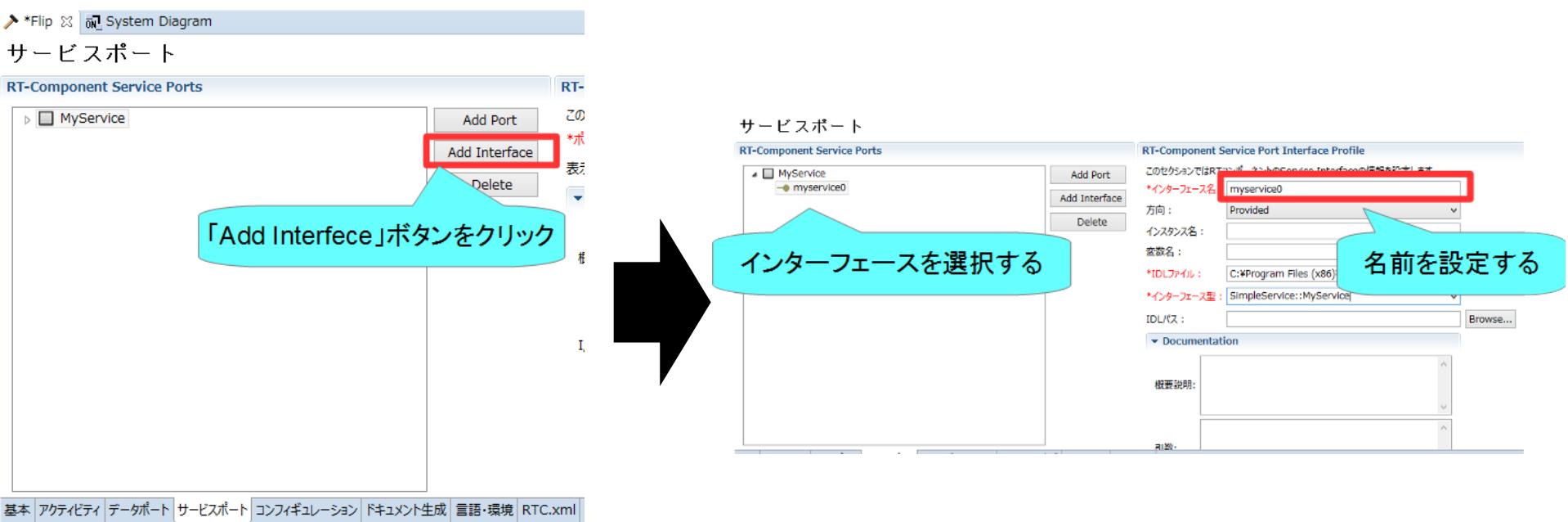
# サービスポートの設定

- サービスポートの追加、インターフェースの追加、設定を行う



# サービスポートの設定

- インターフェースを追加する



# サービスポートの設定

- インターフェースの設定を行う

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

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

\*インターフェース名 : myservice0

方向 : Provided

インスタンス名 :

変数名 :

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

\*インターフェース型 : SimpleService::MyService

IDLパス :

インターフェース型を選択

「Browse...」をクリックしてIDLファイルを選択

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

The diagram illustrates the relationship between service interface types and their providers/consumers. It shows two main components: 'Required Interface' (Consumer) and 'Provided Interface' (Provider). The consumer side is represented by a socket icon with a dashed line, labeled '機能を使う側' (User side). The provider side is represented by a port icon with a dashed line, labeled '機能を提供する側' (Provider side). Arrows point from the text labels to their corresponding icons.

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



2016/07/03 18:07 Windows J

2016/07/03 18:07 SH ファイル

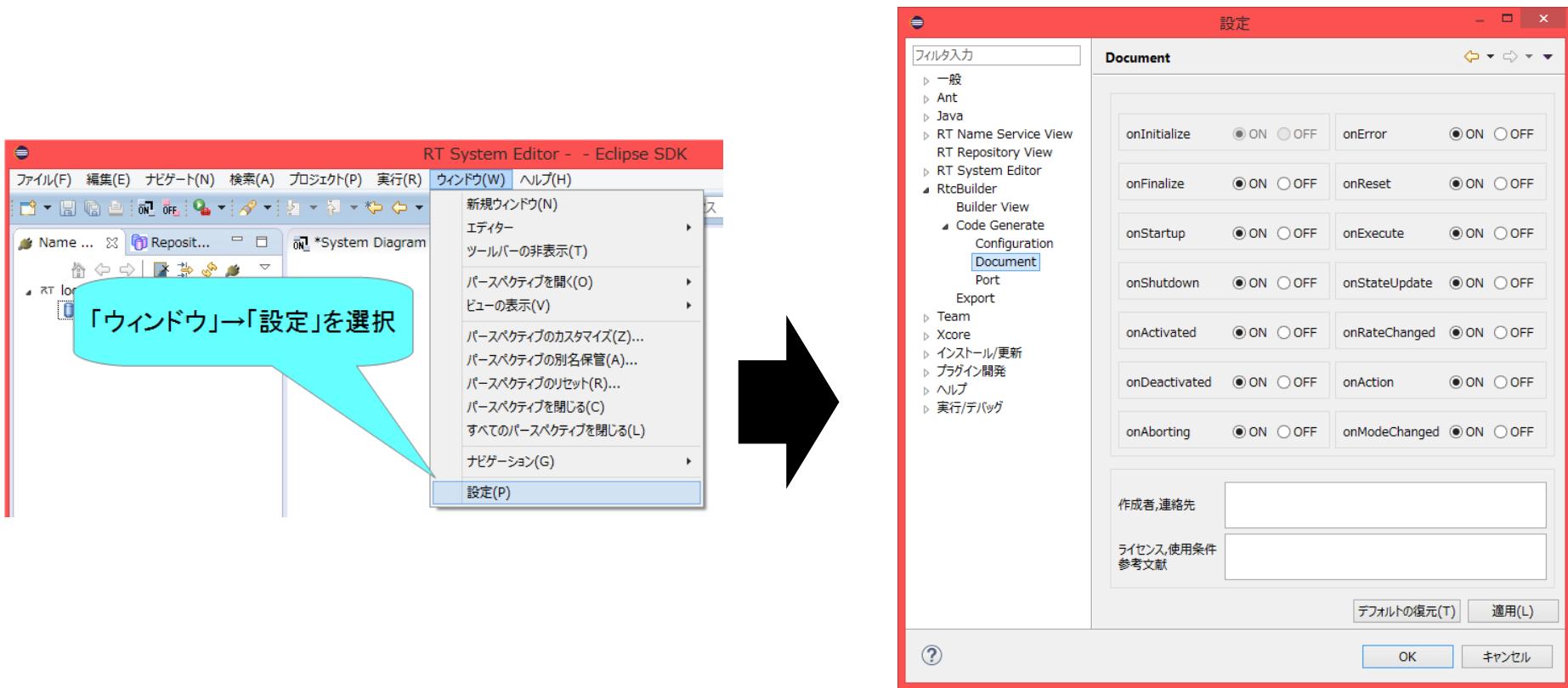
# サービスポートの設定

- IDLファイルについて
  - プログラミング言語に非依存のインターフェース定義言語

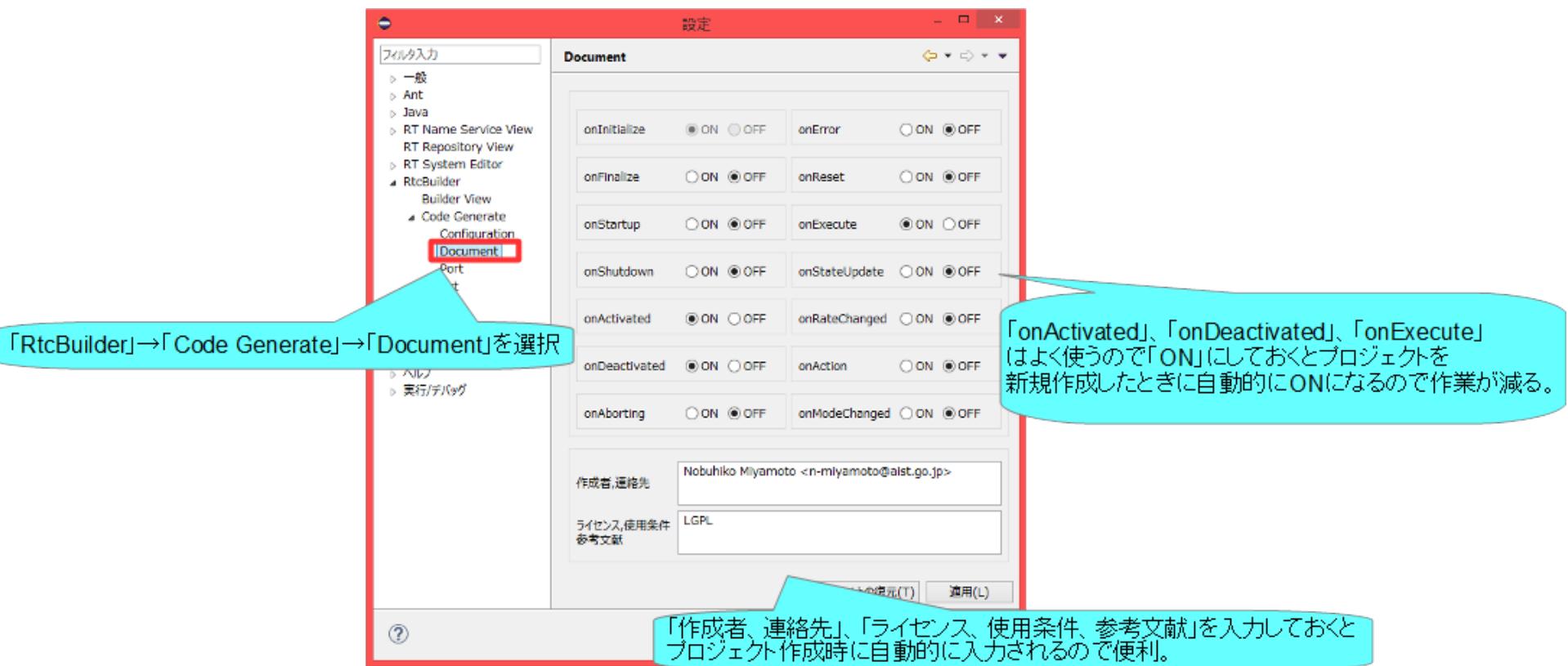
```
1 module SimpleService {↓
2     typedef sequence<string> EchoList;↓
3     typedef sequence<float> ValueList;↓
4     interface MyService↓
5     {↓
6         string echo(in string msg);↓
7         EchoList get_echo_history();↓
8         void set_value(in float value);↓
9         float get_value();↓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

- コンシュマー側でプロバイダ側のecho、get\_valueなどのオペレーションを呼び出す

# RTC Builderに関する設定



# RTC Builderに関する設定



# 独自のデータ型の利用

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

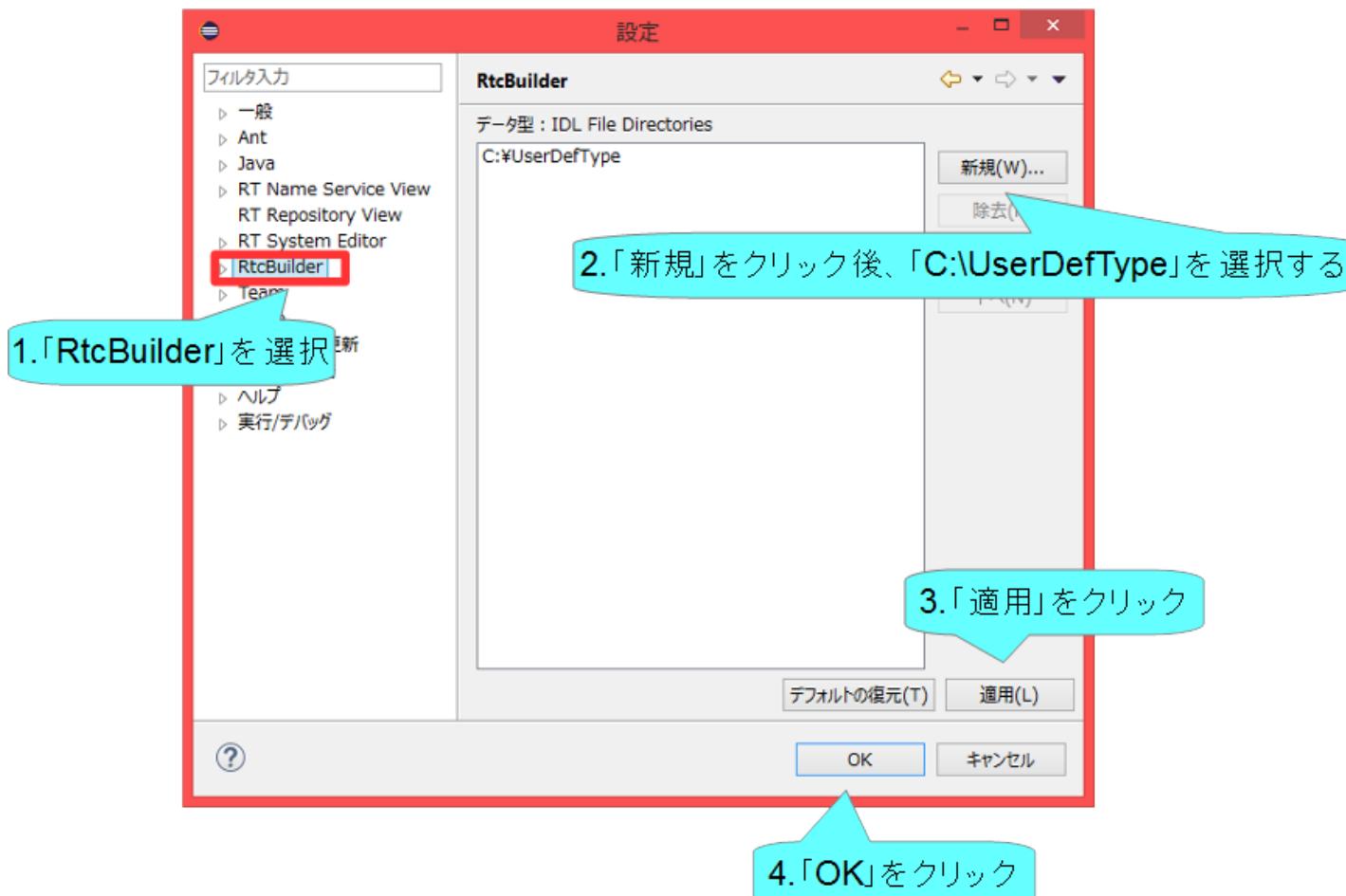
```
1 // @file MyDataType.idl↓
2 #include "BasicDataType.idl"↓
3 ↓
4 struct MyData↓
5 {↓
6     RTC::Time tm;↓
7     short shortVariable;↓
8     long longVariable;↓
9     sequence<double> data;↓
10 } ;[EOF]
```

- 別のIDLファイルをインクルードしている場合は同じフォルダにコピーする

Windows 8... > UserDefType			UserDefTypeの検索
名前	更新日時	種類	
BasicDataType.idl	2014/08/28 20:06	IDL ファイル	
MyDataType.idl	2016/07/03 18:57	IDL ファイル	

# 独自のデータ型の利用

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



# 独自のデータ型の利用

- 独自のデータ型でデータポートの通信を行う手順

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

*ポート名 (InPort)	in	Add	*ポート名 (OutPort)	out	Add
< >		Delete	< >		Delete

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

ポート名 :

\*データ型

変数名

表示位置

Document

Document

Document

デーティ型一覧にMyDataが追加

# RT System Editor

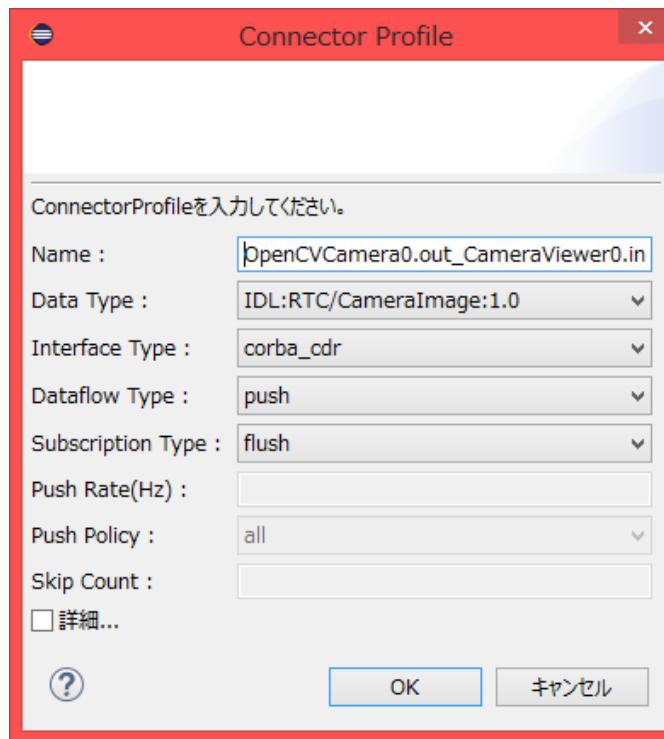
## 補足

# コネクタプロファイルの設定

項目	設定内容
Name	接続の名称
DataType	ポート間で送受信するデータの型. ex)TimedOctet, TimedShortなど
InterfaceType	データを送信方法. ex)corba_cdrなど
DataFlowType	データの送信手順. 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

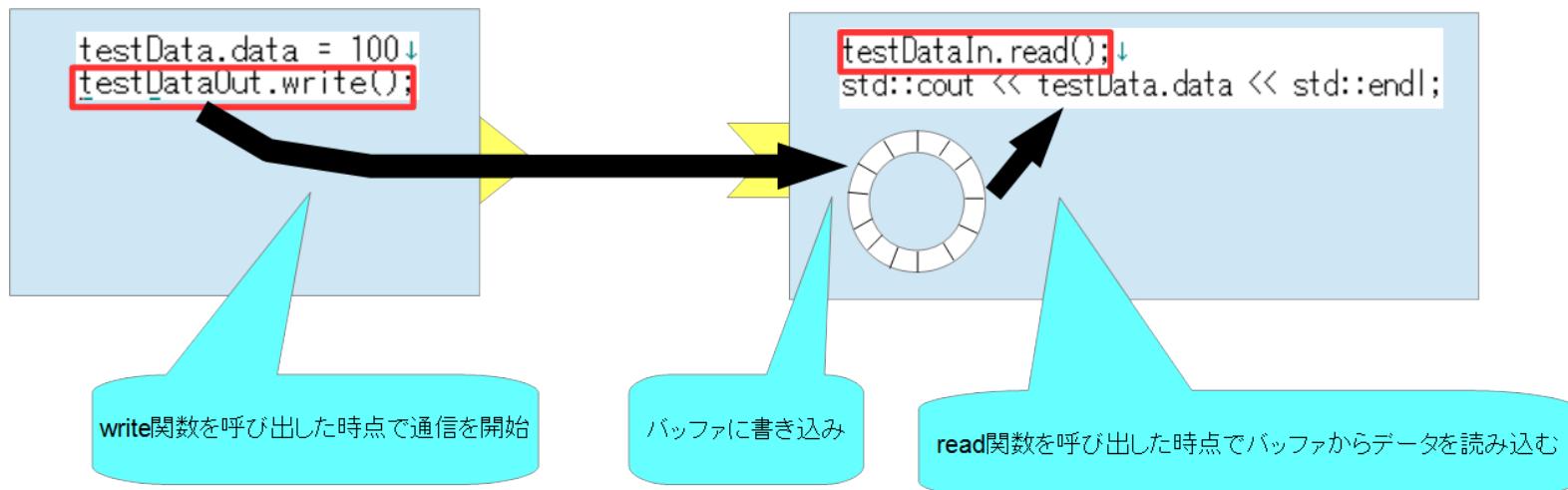
# コネクタプロファイルの設定

- InterfaceType
  - データの送信方法
  - 1.1.2ではcorba\_cdr(CORBAによる通信)のみ選択可能
  - 1.2.0では以下の通信方法も選択可能になる予定
    - direct(同一プロセスで起動したRTC間でデータを直接変数に渡す)
    - shared\_memory(共有メモリによる通信)
- DataFlowType
  - データの送信手順
    - Push
      - OutPortがInPortにデータを送る
    - Pull
      - InPortがOutPortに問い合わせてデータを受け取る
- SubscriptionType
  - データ送信タイミング(DataFlowTypeがPush型のみ有効)
    - flush(同期)
      - バッファを介さず即座に同期的に送信
    - new(非同期)
      - バッファ内に新規データが格納されたタイミングで送信
    - periodic(非同期)
      - 一定周期で定期的にデータを送信
- Push Policy(SubscriptionTypeがnew、periodicのみ有効)
  - データ送信ポリシー
    - all
      - バッファ内のデータを一括送信
    - fifo
      - バッファ内のデータをFIFOで1個ずつ送信
    - skip
      - バッファ内のデータを間引いて送信
    - new
      - バッファ内のデータの最新値を送信(古い値は捨てられる)

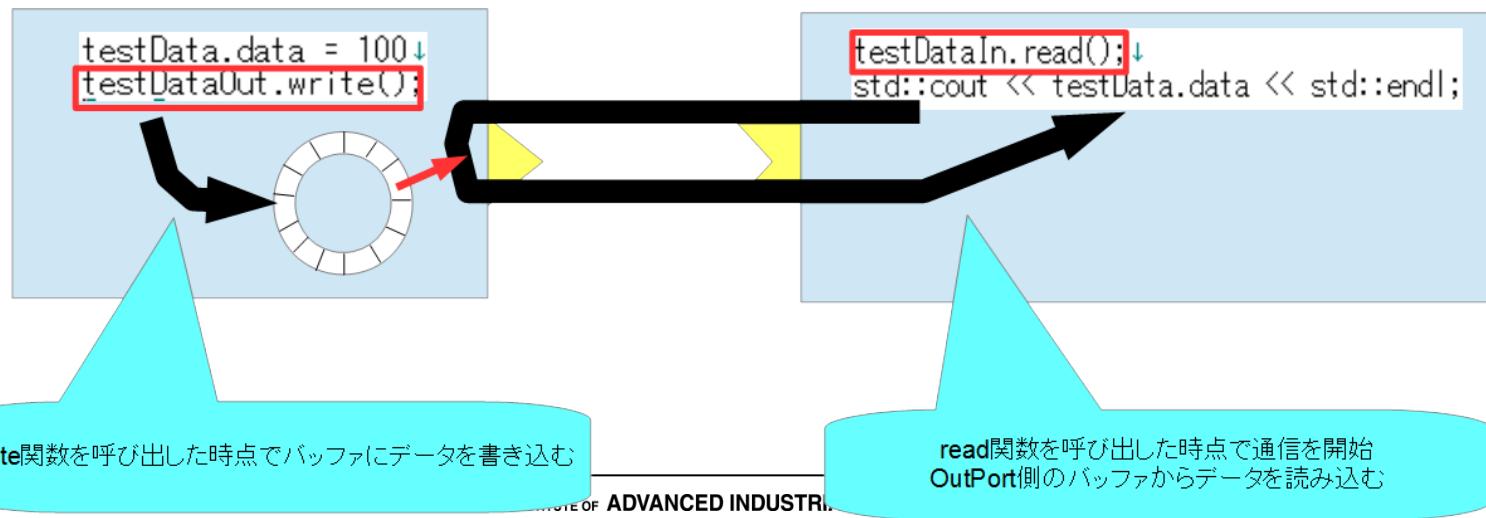


# コネクタプロファイルの設定

- DataFlowType
  - Push



- Pull

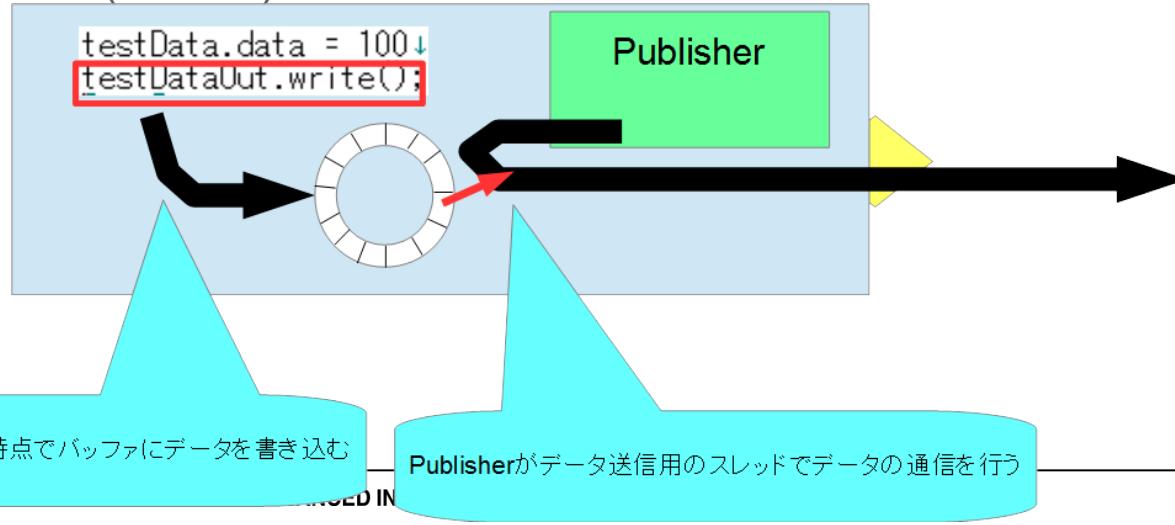


# コネクタプロファイルの設定

- SubscriptionType
  - flush(同期)

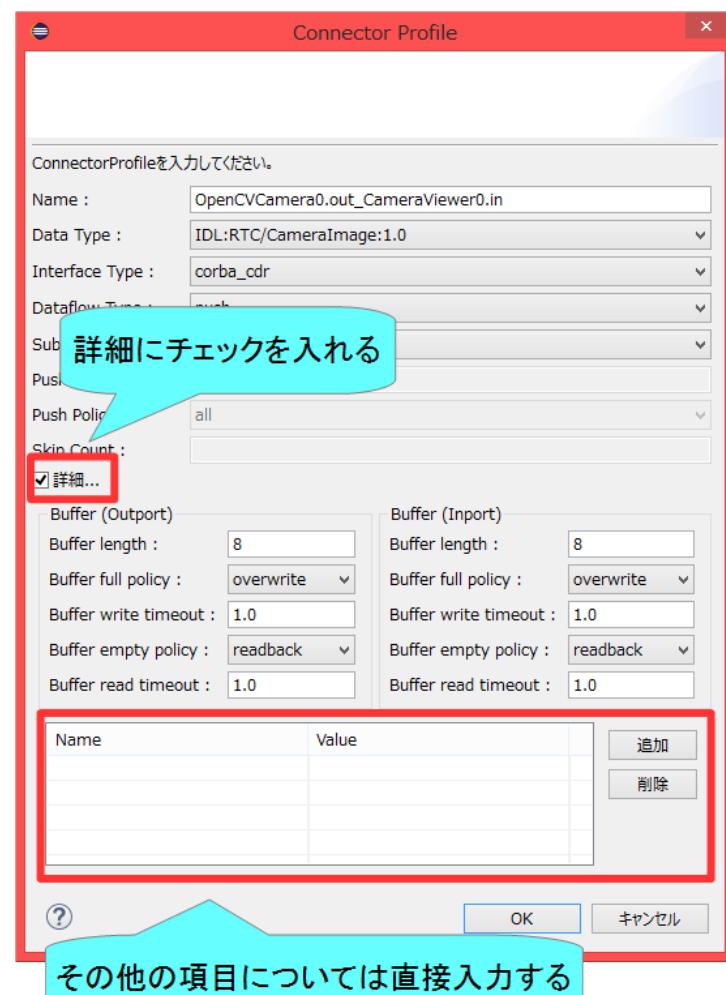


- new、periodic(非同期)



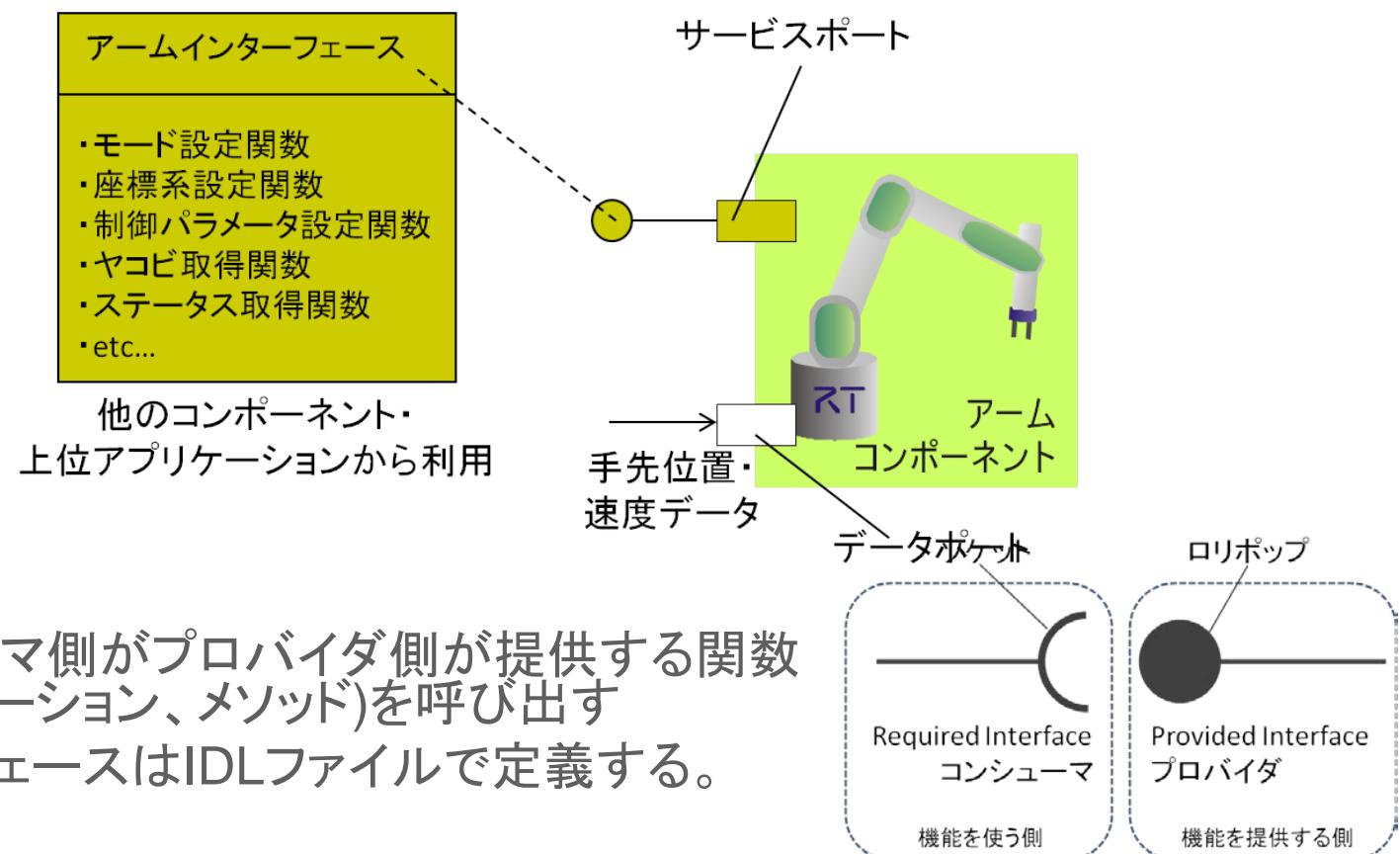
# コネクタプロファイルの設定

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



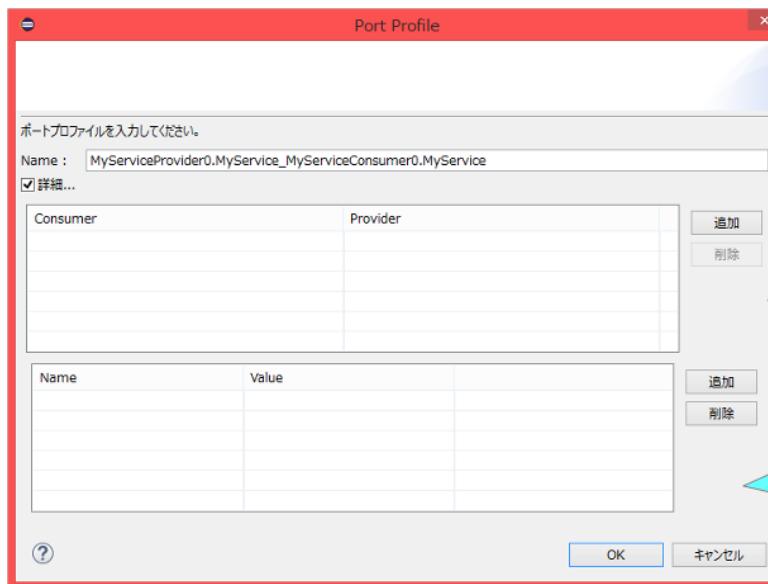
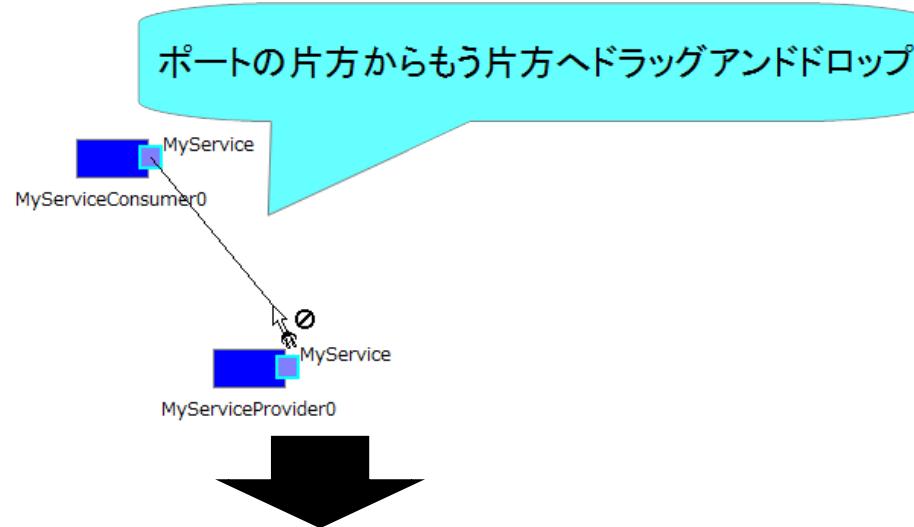
# サービスポートについて

- 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



名前の設定

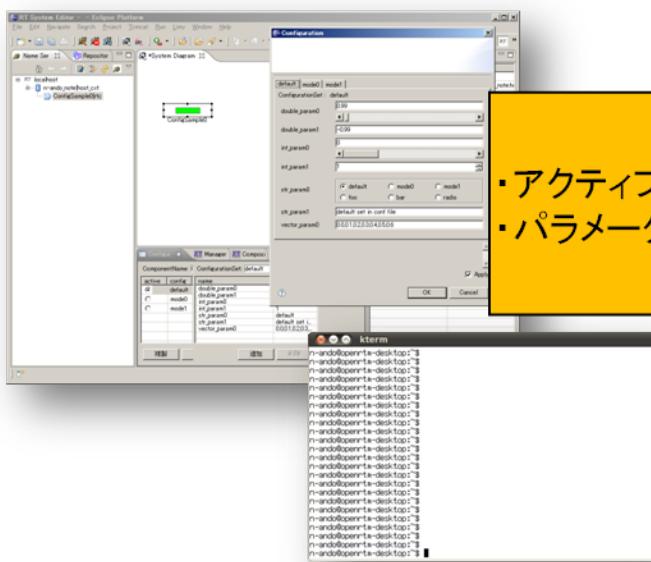
接続するインターフェースの設定  
複数のインターフェースが定義されていた場合に、  
どのインターフェースに接続するかを設定

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

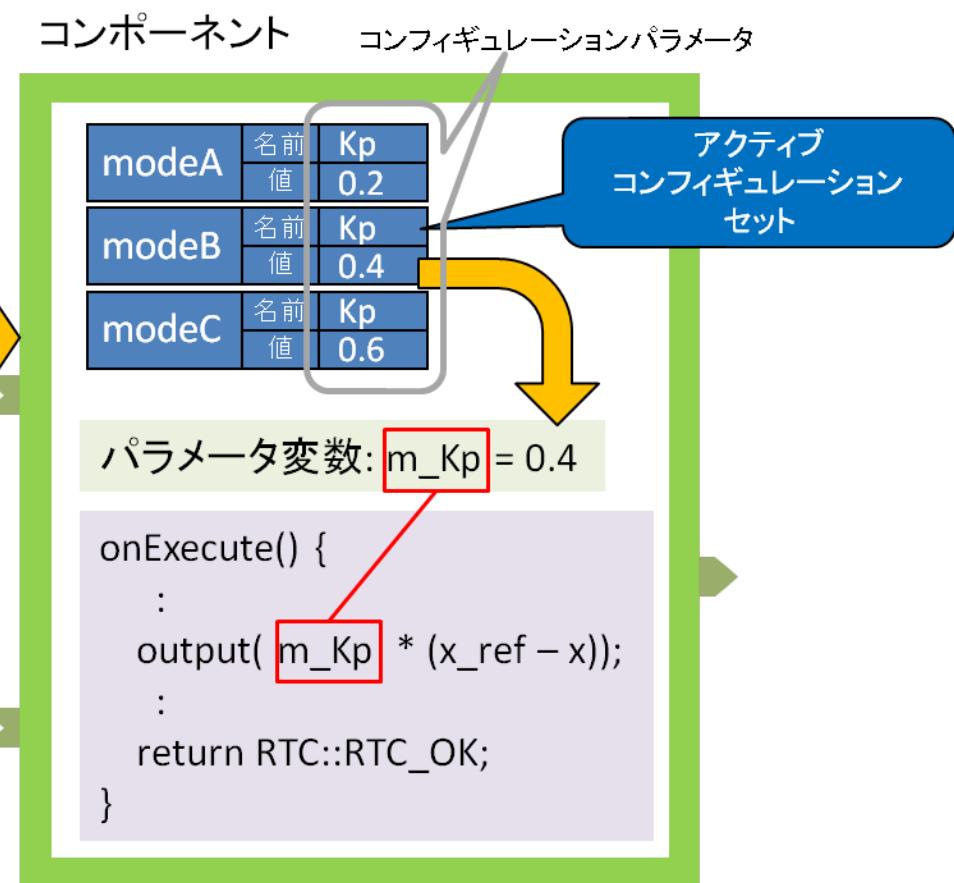
- パラメータを外部から操作する仕組み
  - コンポーネント作成後に変更が必要なパラメータを設定する
    - 例えばデバイスが接続されているCOMポート番号の設定等

## ツール・アプリケーション



- アクティブセットの変更
- パラメータ値の変更

ツール・アプリケーションから、コンポーネント内部で使用する変数の値を変更できる。



# コンフィギュレーションパラメータの設定

対象のRTCをクリックすると表示

「Configuration View」タブを選択

パラメーター一覧

コンフィギュレーションセット一覧

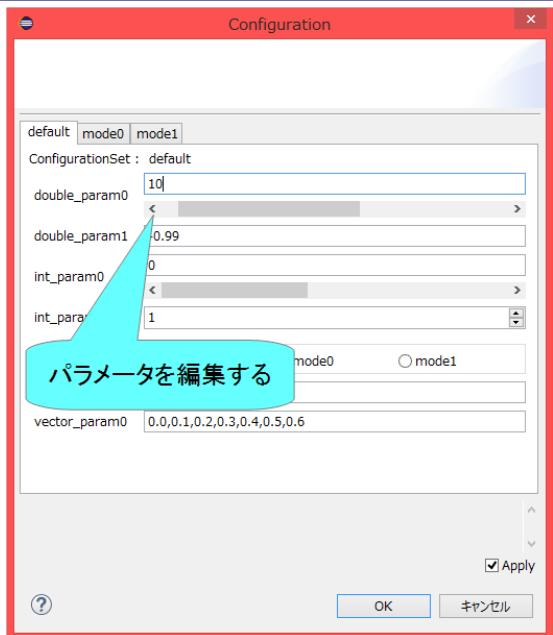
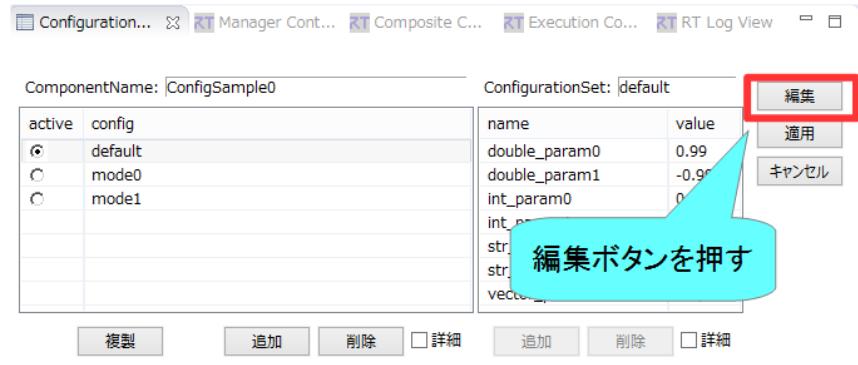
The screenshot shows the RT-Middleware Configuration View interface. At the top, there's a toolbar with tabs: Configuration..., RT Manager Cont..., RT Composite C..., RT Execution Co..., and View. Below the toolbar, the main area has two sections: 'ComponentName: ConfigSample0' and 'ConfigurationSet: default'. The left section contains a table with columns 'active' and 'config', showing three rows: 'default' (selected), 'mode0', and 'mode1'. The right section contains a table with columns 'name' and 'value', listing various parameters like double\_param0 (0.99), double\_param1 (-0.99), int\_param0 (0), int\_param1 (1), str\_param0 (default), str\_param1 (defau...), and vector\_param0 (0.0,0,...). On the far right, there are three buttons: '編集' (Edit), '適用' (Apply), and 'キャンセル' (Cancel). A large blue callout at the bottom left points to the 'ComponentName' field and is labeled 'コンフィギュレーションセット一覧' (Configuration Set List). A blue callout at the top right points to the 'ConfigurationSet' dropdown and is labeled 'パラメーター一覧' (Parameter List). A blue callout on the left side points to the 'active' column in the table and is labeled '対象のRTCをクリックすると表示' (Click on the target RTC to display). A blue callout at the bottom left points to the 'active' column in the table and is labeled '「Configuration View」タブを選択' (Select the 'Configuration View' tab).

active	config
<input checked="" type="radio"/>	default
<input type="radio"/>	mode0
<input type="radio"/>	mode1

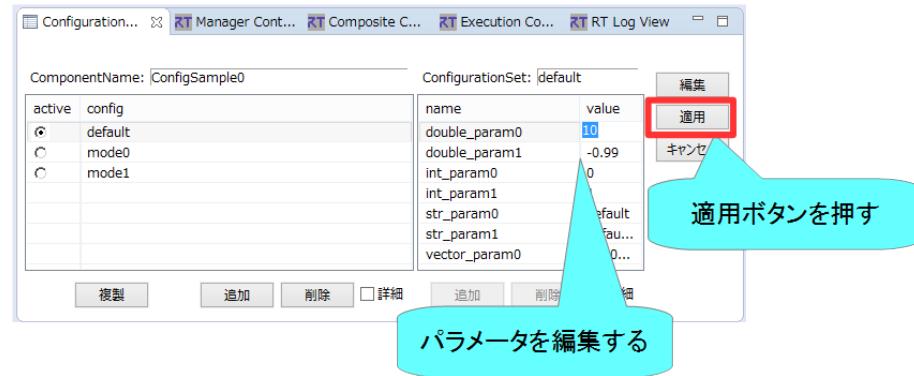
name	value
double_param0	0.99
double_param1	-0.99
int_param0	0
int_param1	1
str_param0	default
str_param1	defau...
vector_param0	0.0,0...

# コンフィギュレーションパラメータの設定

- 方法1

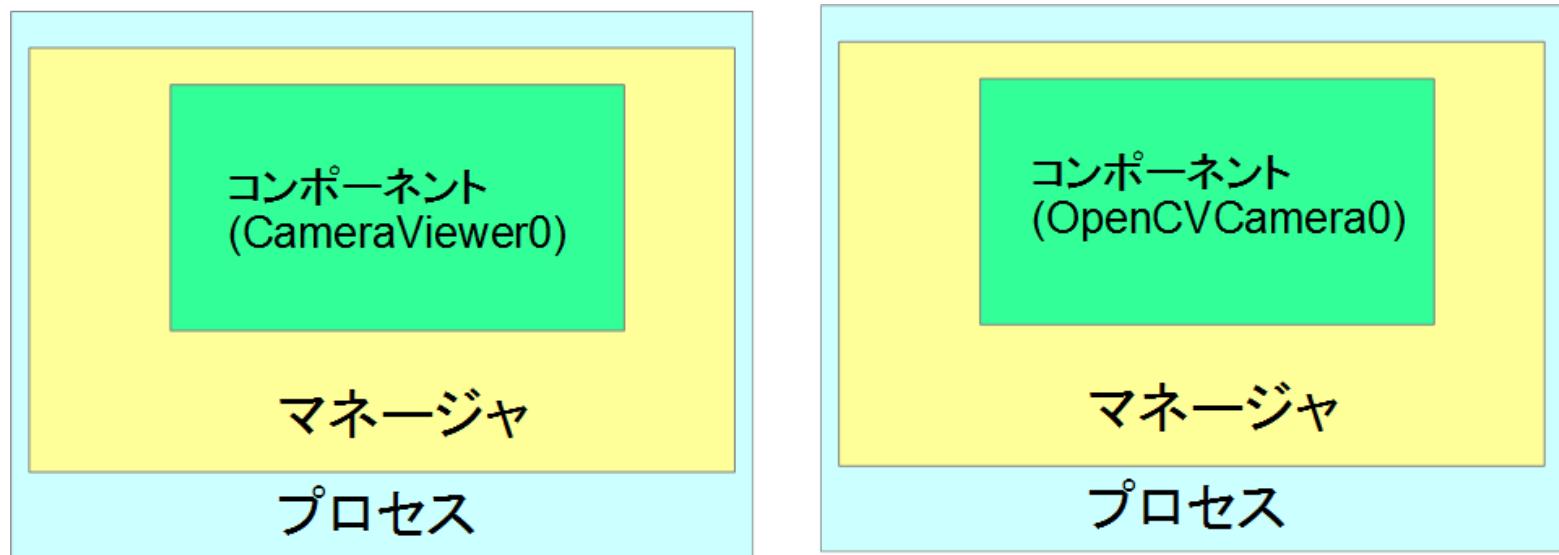


- 方法2



# 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

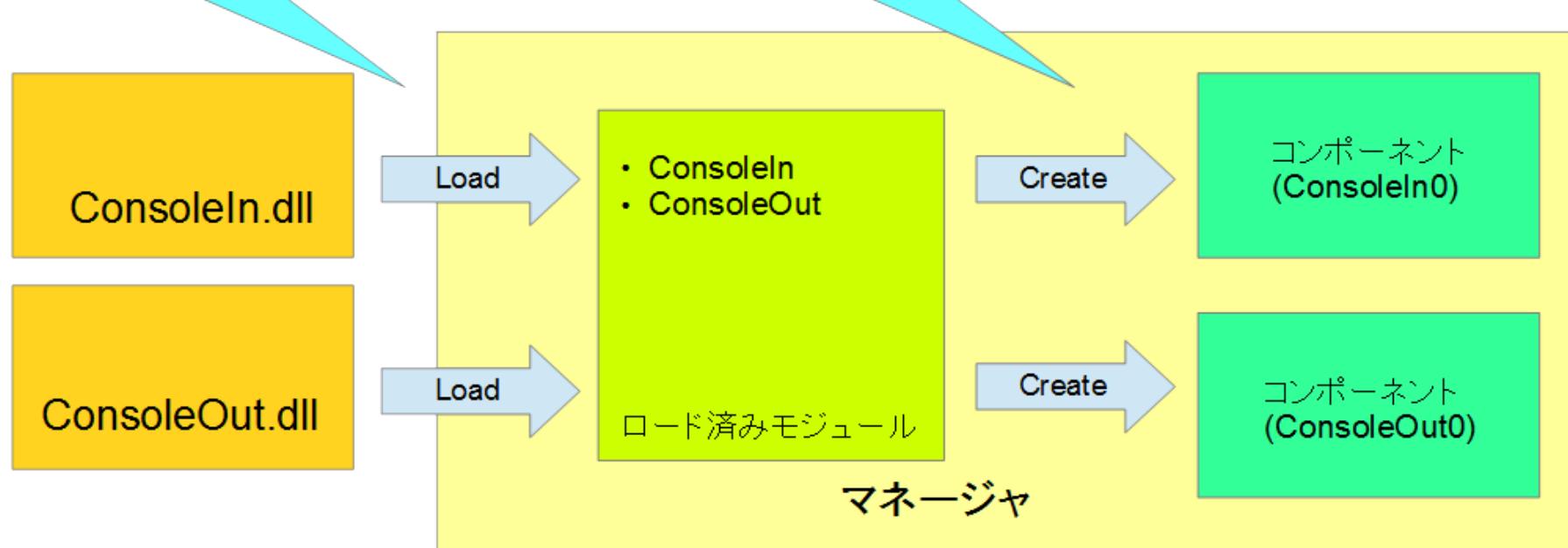


基本的にマネージャは各プロセスに1つ起動する。  
マネージャがコンポーネントを起動する

# マネージャの操作

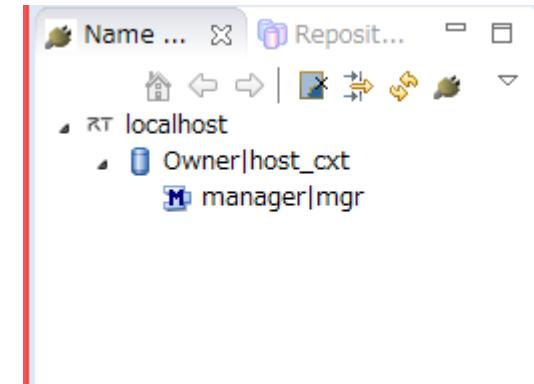
1. モジュールをロードする
  - ・C++:.dll、.so
  - ・Python:.py
  - ・Java:.jar

2. コンポーネントを起動する



# マネージャの操作

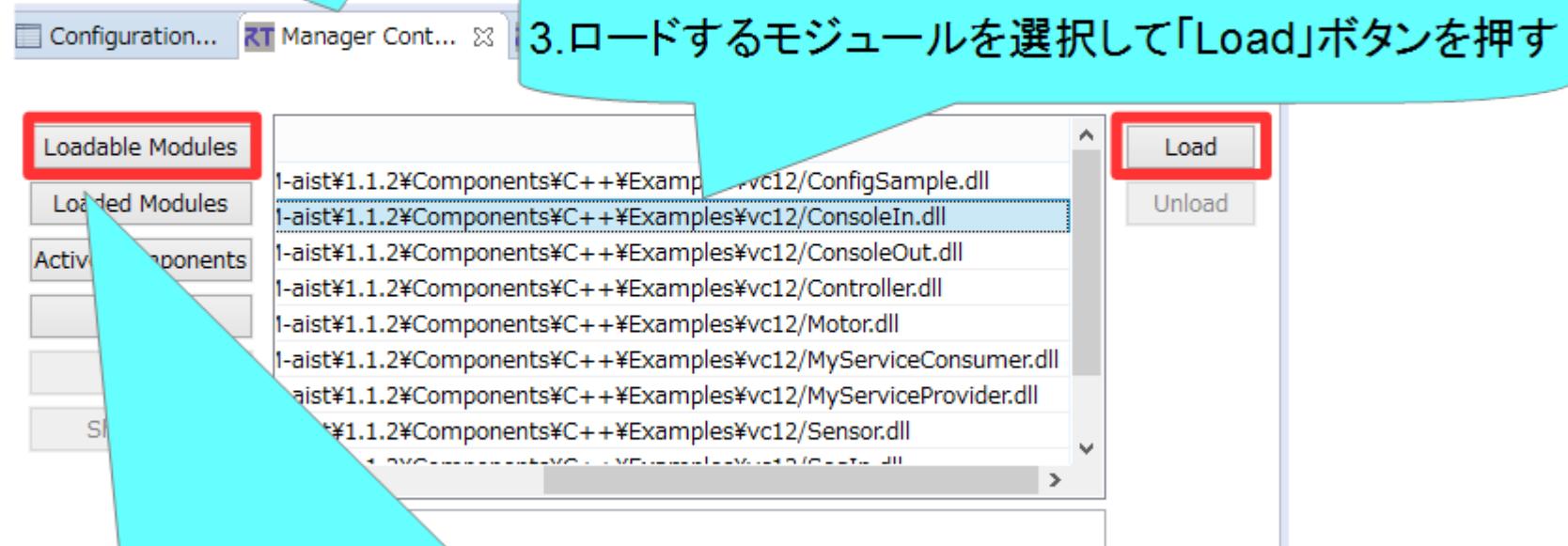
- マスターマネージャの起動、RT System Editorからの操作によるRTCの生成までの手順を説明する
  - rtc.confの設定
    - 「manager.is\_master」を「YES」に設定して起動するマネージャをマスターに設定する
      - manager.is\_master: YES
    - モジュール探索パスの設定
      - manager.modules.load\_path: .., C:¥¥Program Files (x86)¥¥OpenRTM-aist¥¥1.1.2¥¥Components¥¥C++¥¥Examples¥¥vc12
  - 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 System Editor の ネームサービス ビュー に マネージャ が 表示される



# マネージャの操作

- モジュールのロード

1.「Manager Control View」タブを選択

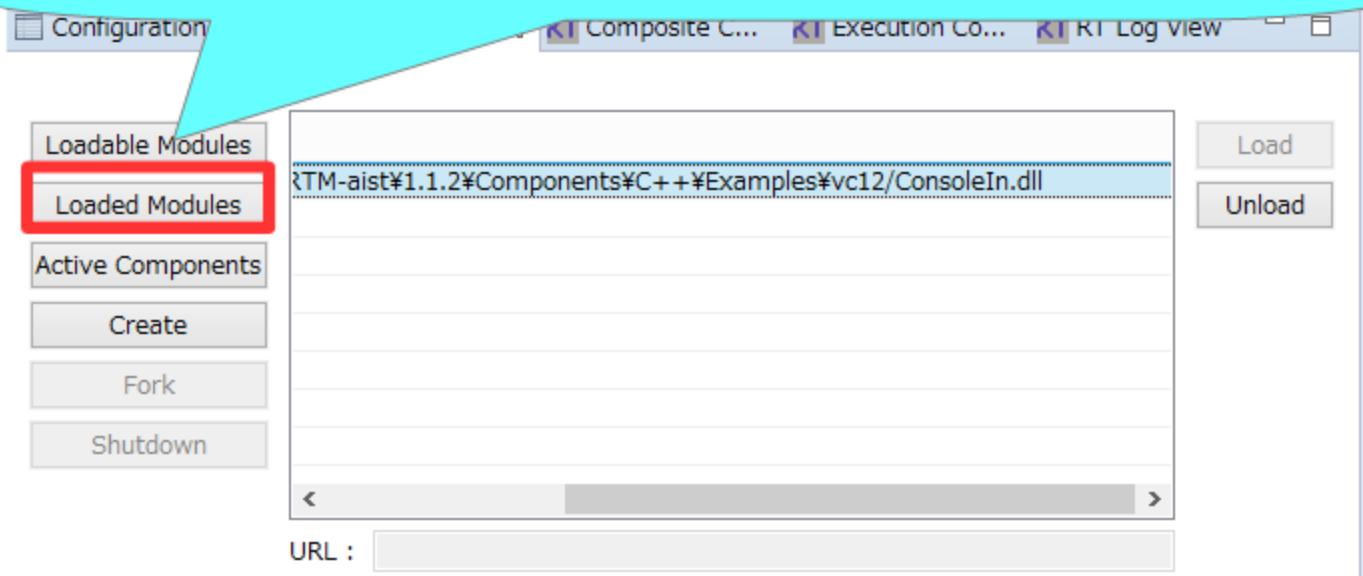


2.「Loadable Modules」ボタンを押すとロード可能なモジュール一覧表示

# マネージャの操作

- モジュールのロード

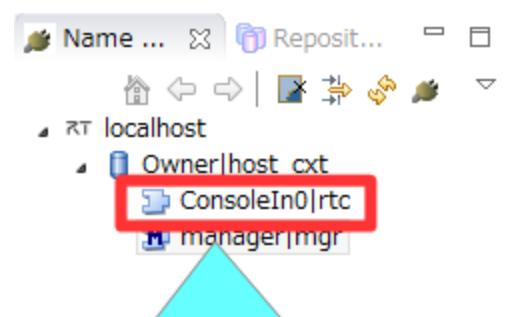
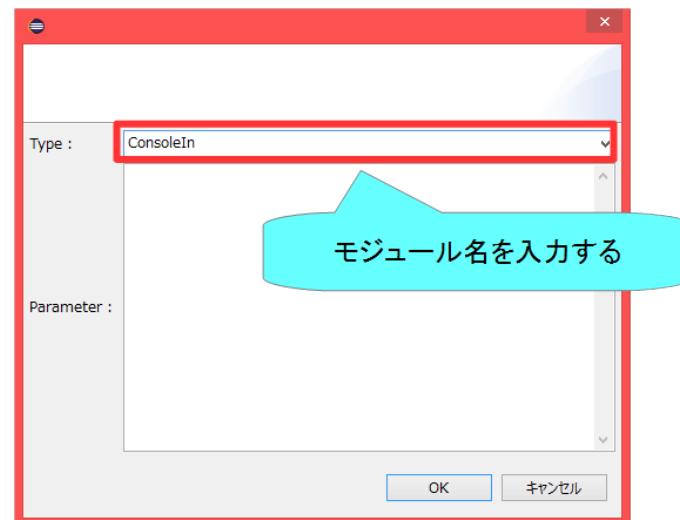
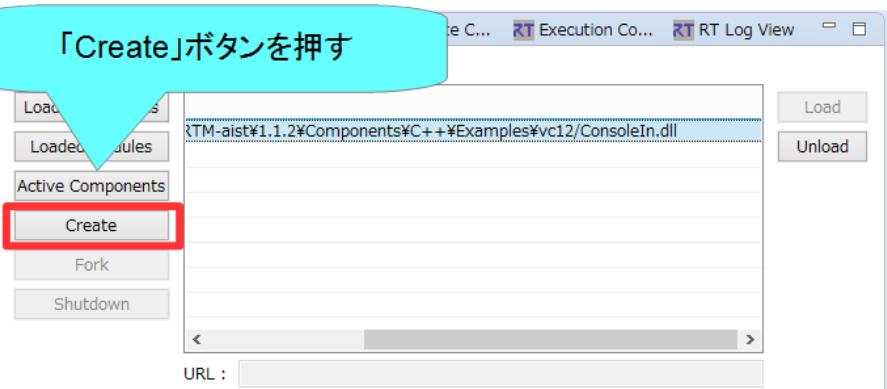
「Loaded Modules」ボタンを押すとロード済みのモジュール一覧表示



# マネージャの操作

- RTCの生成

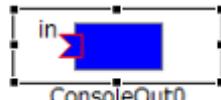
「Create」ボタンを押す



指定したRTCが起動する

# 実行コンテキストの操作

RTCをクリック



「Execution Context View」タブを選択

実行周期

The screenshot shows the RT Execution Context View window with the following details:

component: ConsoleOut0

Execution Context (Table):

owned0
--------

rate: 1000.0

Execution Context (Table):

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

Buttons on the right:

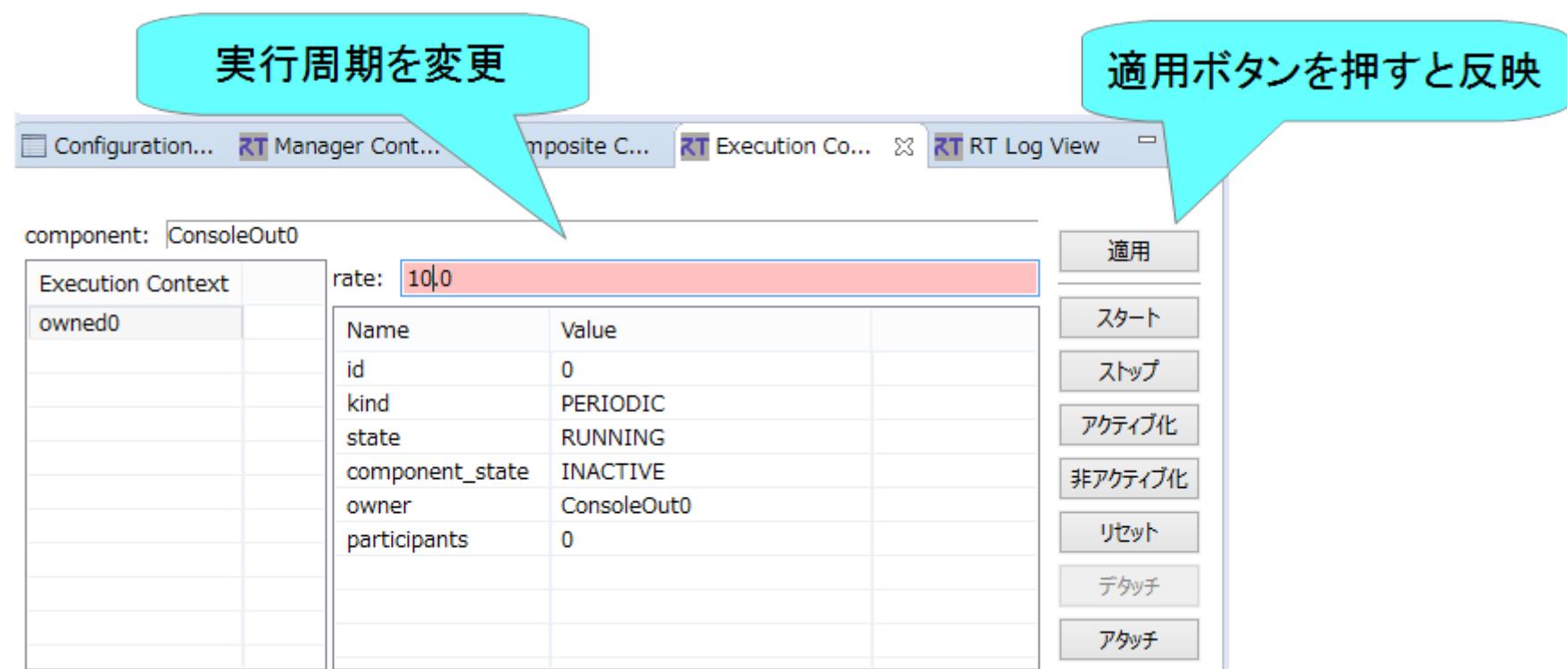
- 適用 (Apply)
- スタート (Start)
- ストップ (Stop)
- アクティビ化 (Activate)
- 非アクティビ化 (Deactivate)
- リセット (Reset)
- デタッチ (Detach)
- アタッチ (Attach)

関連付けている実行コンテキスト一覧

実行コンテキストの情報

# 実行コンテキストの操作

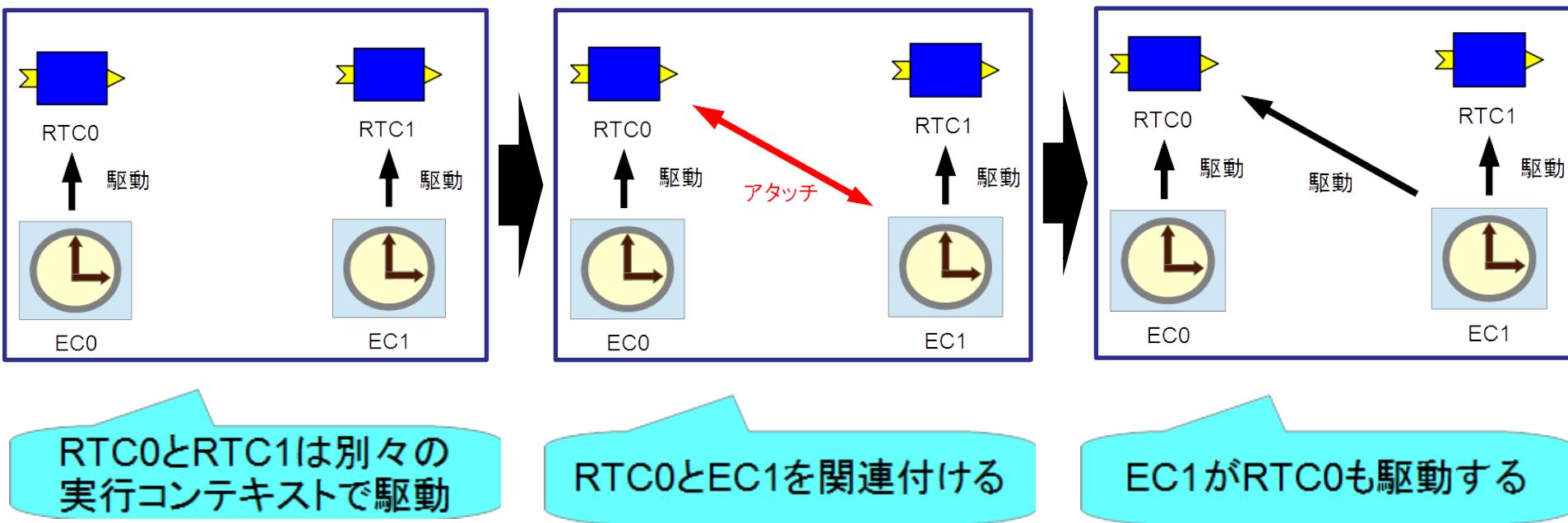
- 実行周期の設定



# 実行コンテキストの操作

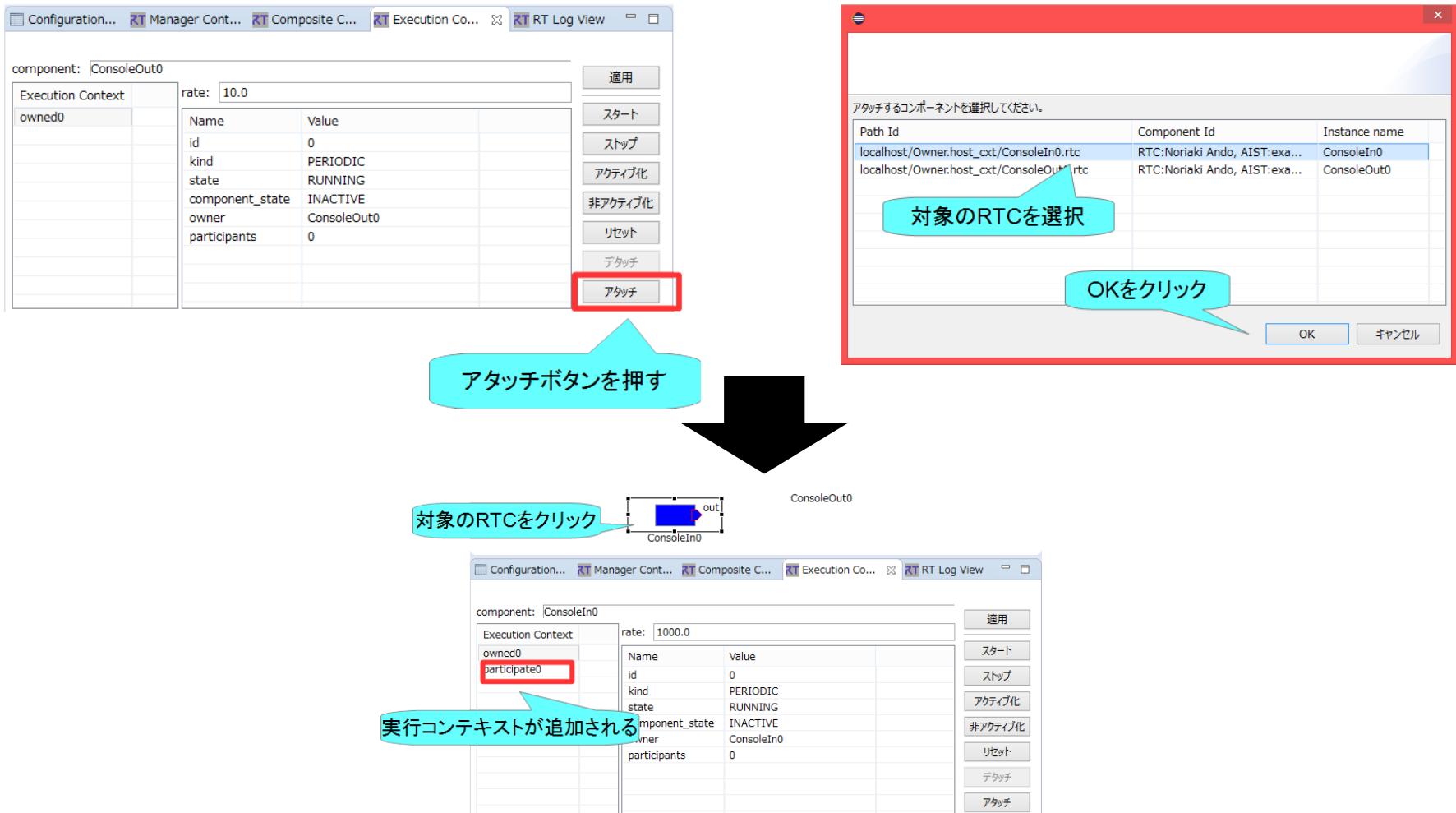
- 実行コンテキストの関連付け

- 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



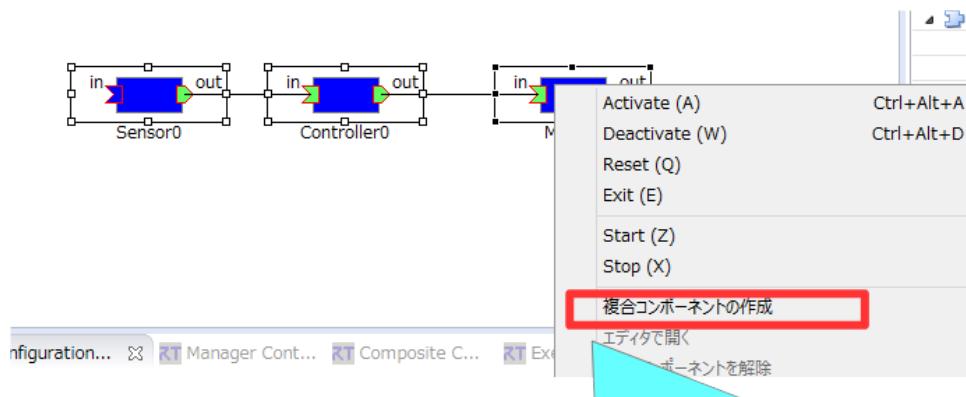
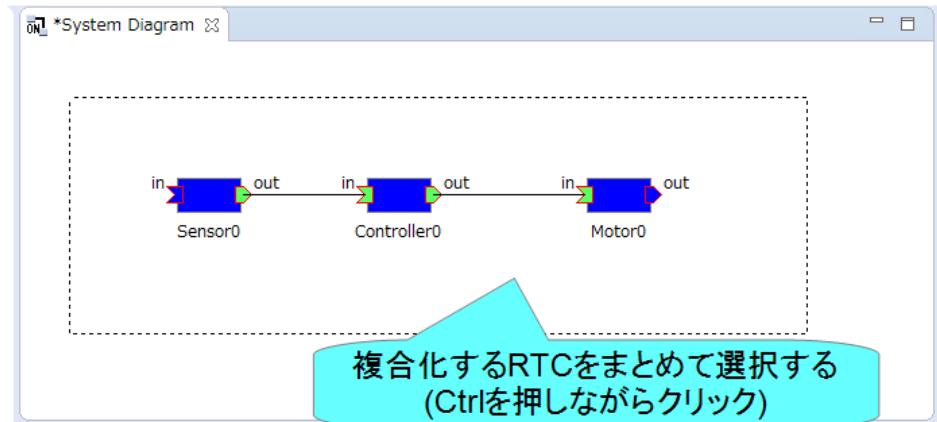
# 実行コンテキストの操作

- 実行コンテキストの関連付け



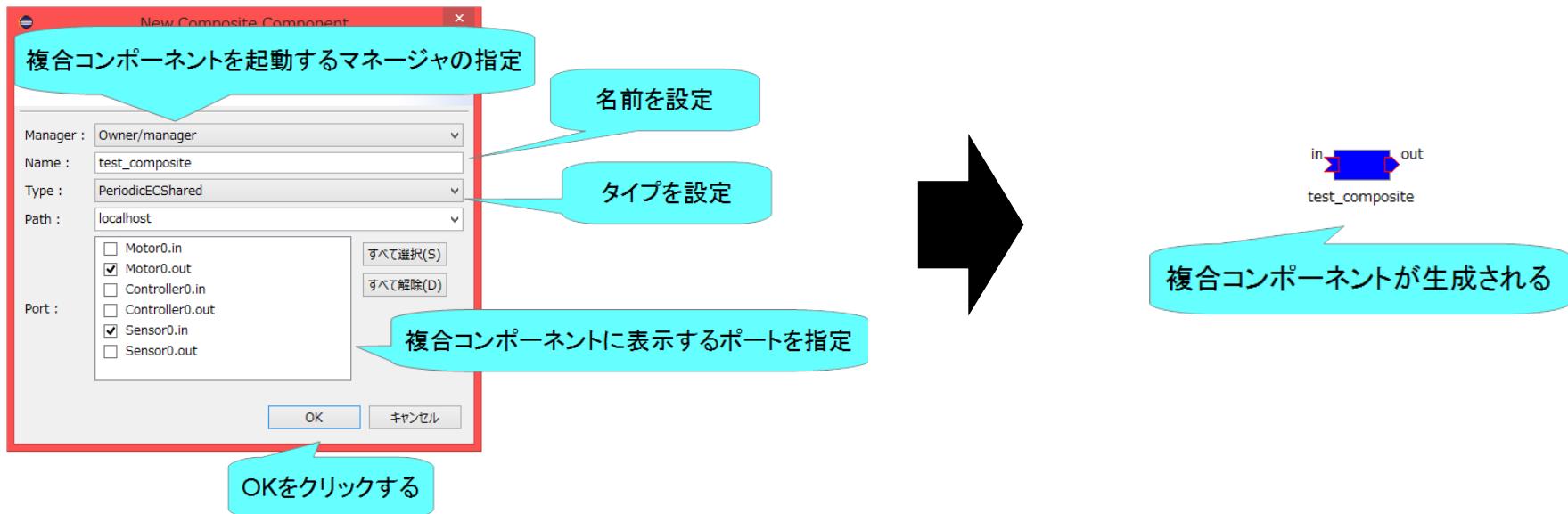
# 複合コンポーネントの操作

- 複合コンポーネントの生成



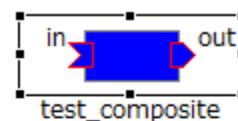
# 複合コンポーネントの操作

- 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をクリック

「Composite Component View」タブを選択

Screenshot of the RT Middleware Composite Component View interface. The top navigation bar includes Configuration..., Manager Cont..., Composite C..., Execution Co..., RT Log View, and other tabs. Below the bar, there are input fields for "component: test\_composite" and "type: PeriodicECShared". A table lists port mappings:

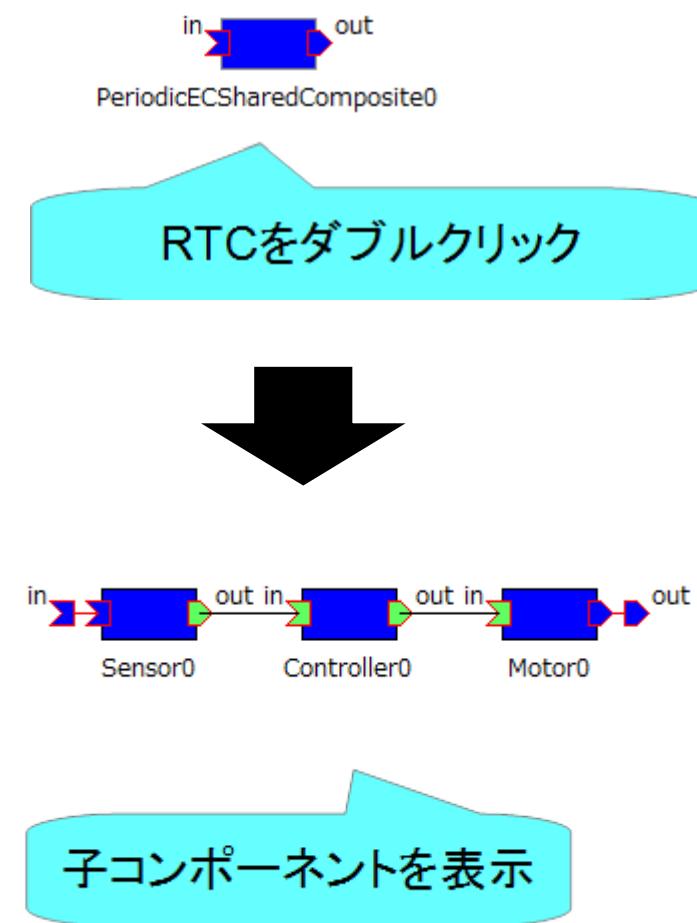
	component	port
<input type="checkbox"/>	Controller0	Controller0.in
<input type="checkbox"/>	Controller0	Controller0.out
<input checked="" type="checkbox"/>	Sensor0	Sensor0.in
<input type="checkbox"/>	Sensor0	Sensor0.out
<input type="checkbox"/>	Motor0	Motor0.in
<input checked="" type="checkbox"/>	Motor0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

On the right side of the table, there are "適用" (Apply) and "キャンセル" (Cancel) buttons. A large cyan callout points to the "適用" button with the text "適用ボタンを押すと変更を反映". Another cyan callout at the bottom left points to the table with the text "表示するポートの選択".

表示するポートの選択

適用ボタンを押すと変更を反映

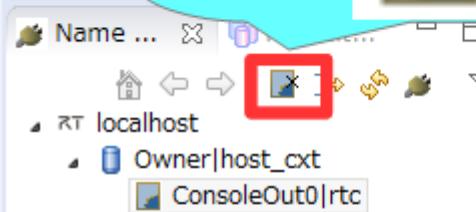
# 複合コンポーネントの操作



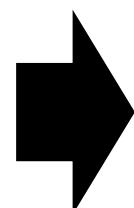
# ゾンビの削除

- 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

ゾンビクリアボタンを押す

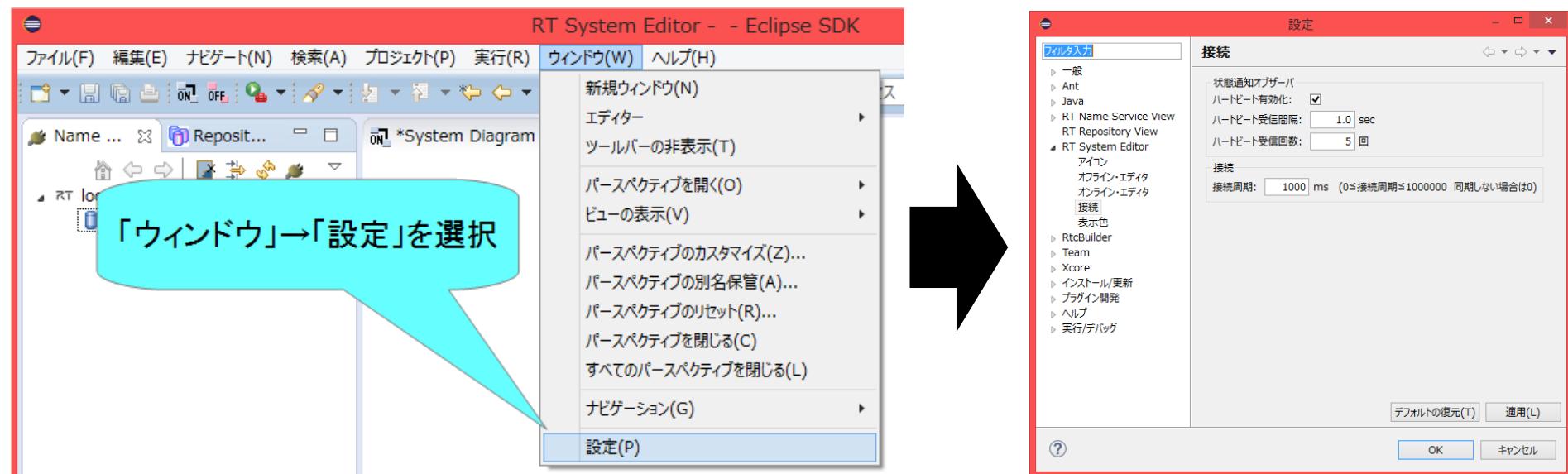


ゾンビ



ゾンビが消える

# RT System Editorに関する設定



# How to check the progress

- Check with Zoom's "Raise Hand Button"

Display the participant's screen.

ミュート解除 ビデオの開始 参加者 画面を共有 レコーディング 反応 退出

ブレークアウト セッションの参加者 (1)

test (自分)	手を上げる	手を下げる
-----------	-------	-------

Raise your hand if there is a problem

If it is resolved, lower your hand

ミュートを解除します 手を挙げる 会議ウィンドウにマージ ミュートを解除します 手を下げる 会議ウィンドウにマージ RIA

124