

SICE 2011

RT-Middleware Tutorial

Date : 2011/9/13 10:00~16:30

Place : SICE 2011



RT-Middleware tutorial



10:00 - 10:45	Part 1: Introducing RT-Middleware
	Tetsuo Kotoku (AIST)
	An introduction to RT-Middleware, RT-Systems and RT-Components.
11:00 - 12:30	Part 2: Building RT-Systems using RT-Middleware
	Geoffrey Biggs (AIST)
	Hands-on practice using small samples to construct complete RT-Systems.
13:30 - 15:00	Part 3: Creating RT-Components
	Geoffrey Biggs (AIST)
	Hands-on practice creating RT-Components.
15:15 - 16:00	Part 4: Human interaction with OpenHRI
	Yosuke Matsusaka (AIST)
	A demonstration of RT-Components for human-robot interaction.
16:00 - 16:30	Part 5: Discussion

Part 3: Creating RT-Components

Geoffrey Biggs (AIST)



RTCBUILDER

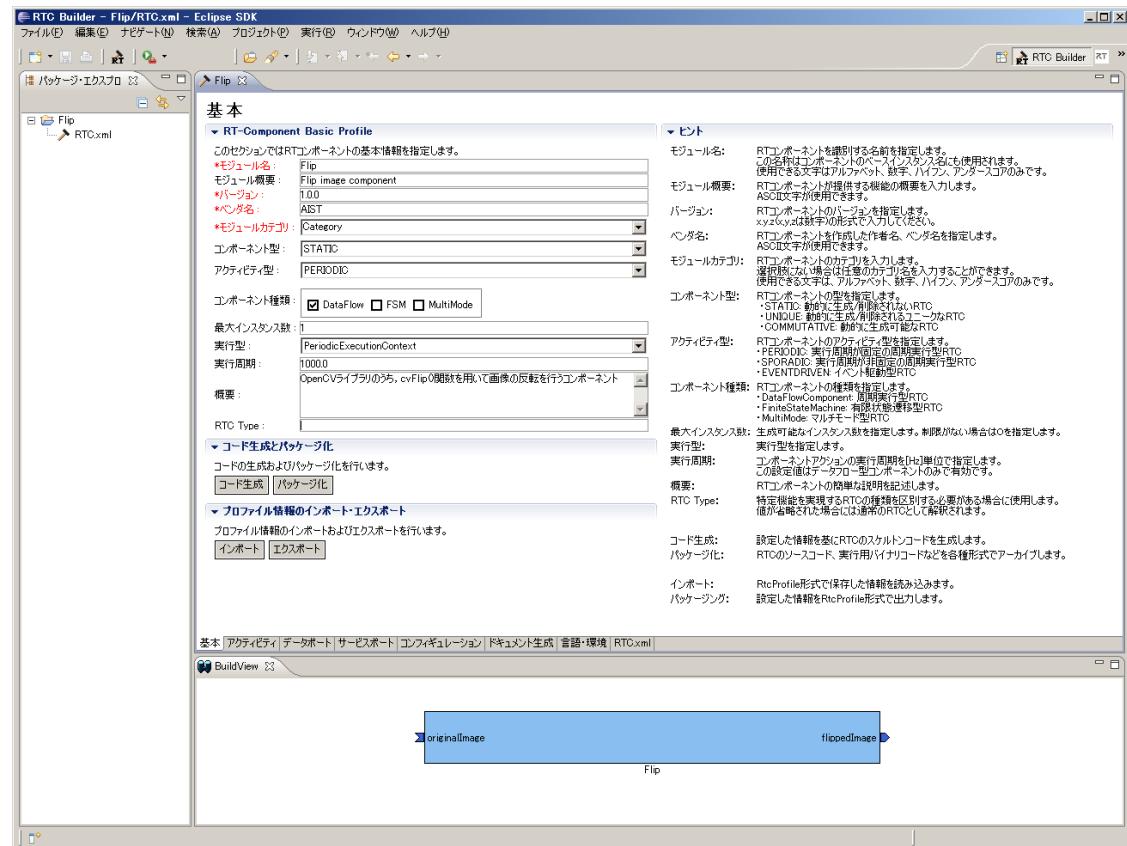


RTCBuilder outline

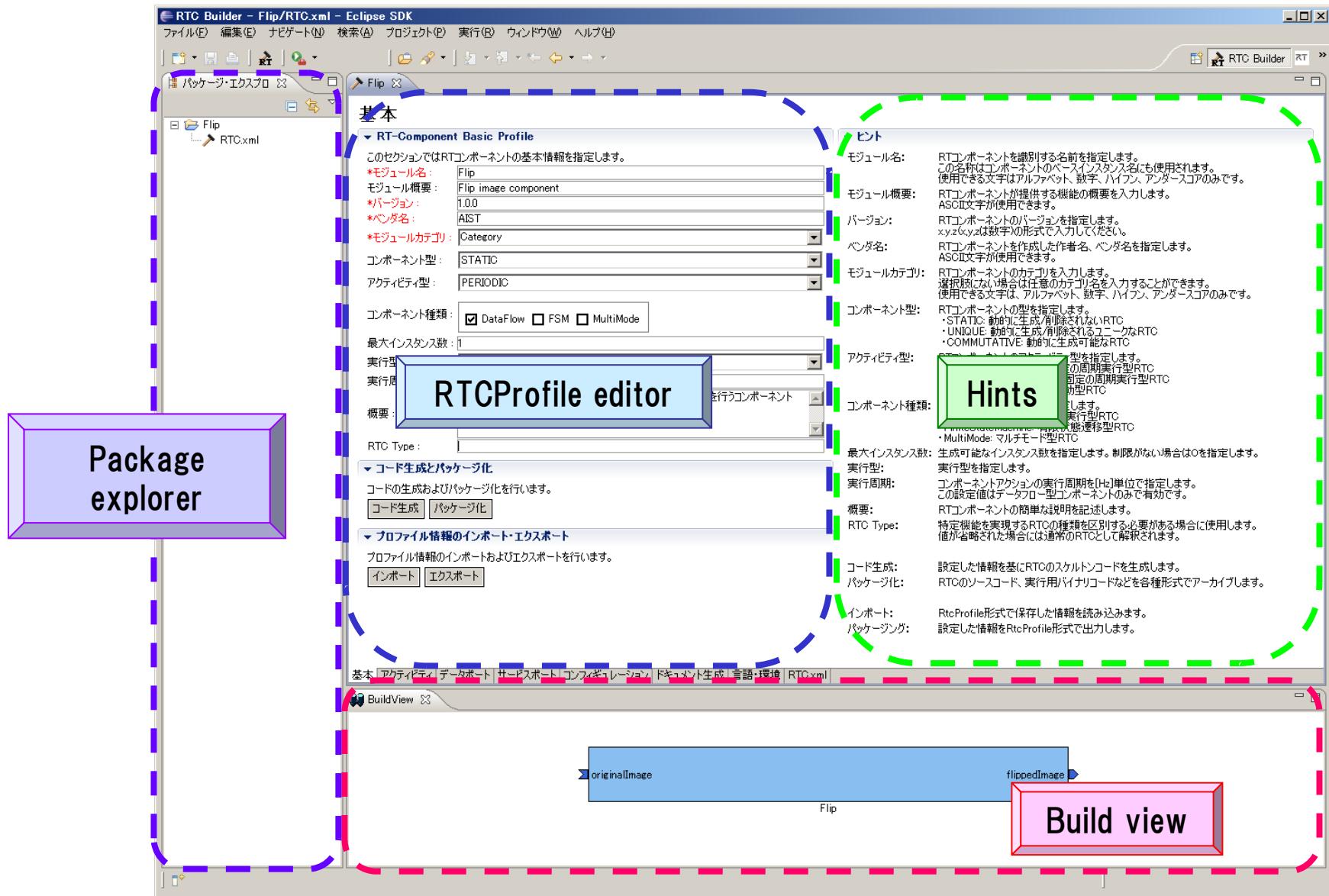
■ What is RTCBuilder?

- A tool for specifying a component profile and generating template source code.
- Support for generating templates in new languages can be added via plugins.
 - C++
 - Java
 - Python

- ✖ C ++ code generation is included by default.
- ✖ Other languages are offered as plugins.



Screen layout



Download the sample component

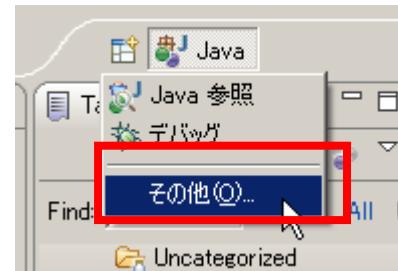
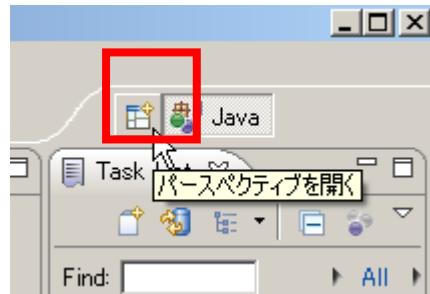


- Download USBCamera.zip from this URL
[http://www.openrtm.org/openrtm/en/content/
sice-2011-openrtm-aist-tutorial](http://www.openrtm.org/openrtm/en/content/sice-2011-openrtm-aist-tutorial)
- Extract the archive.
**※ If extracted to a path with spaces in it, errors will occur
when building with VC++.**

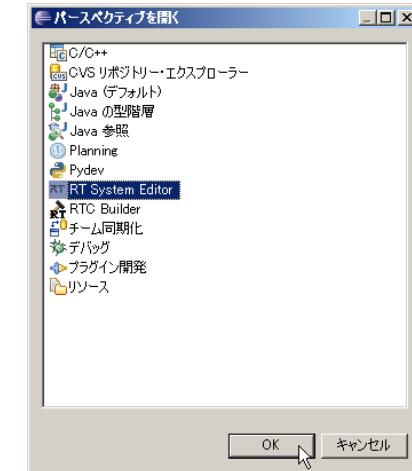
Preparation

■ Change the perspective

- ① Click the “Change perspective” button in the top right, and select “Other”



- ② Select “RTCBuilder”



＊Perspective

A tool in Eclipse.

Changes the menus, toolbars, editors,
views, etc. to match the perspective's
goals.

Project creation and starting the editor

①Click the editor's button in the toolbar



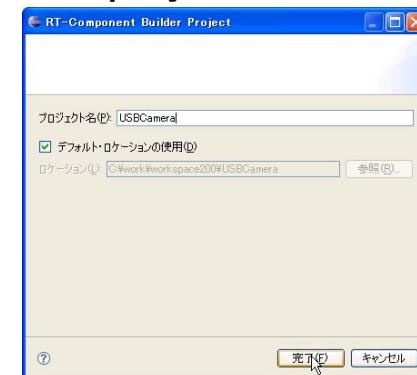
※From the File menu create a new project. In the New Project screen, select “Other” – “RTCBuilder” and click “Next”

※From the File menu, select “Open New Builder Editor”

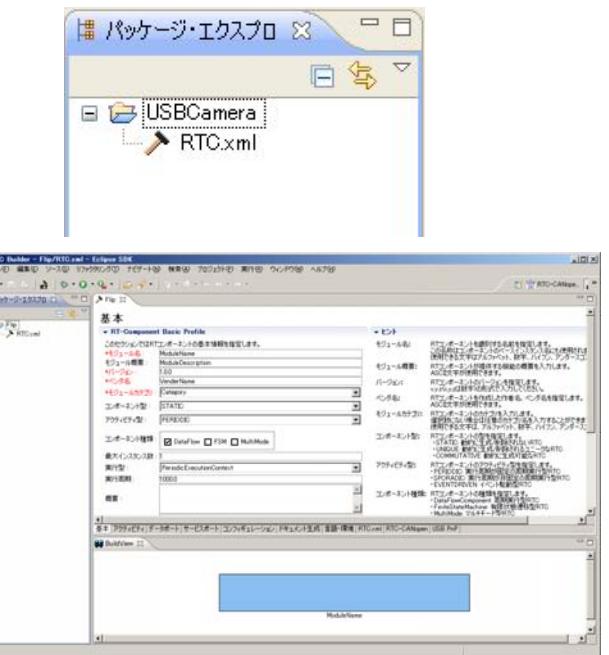
※To specify the project location, in step ②, uncheck the “Default location” checkbox and enter a path. This may be outside your workspace, but it will be treated as if it is inside the workspace.

Project name: USBCamera

②Enter a project name

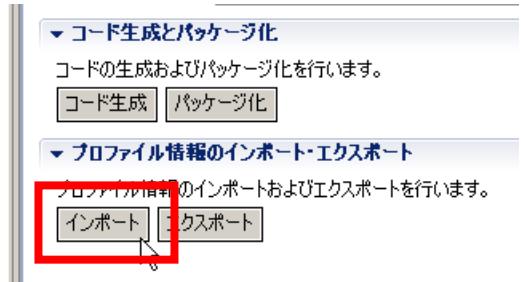


③The project is generated.

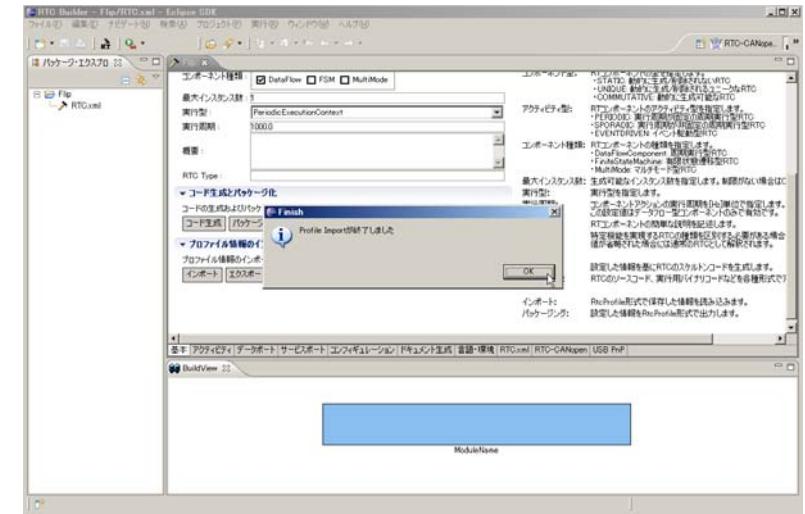


Import the profile

① Click “Import” in the “Basic” tab



② Select the XML file.

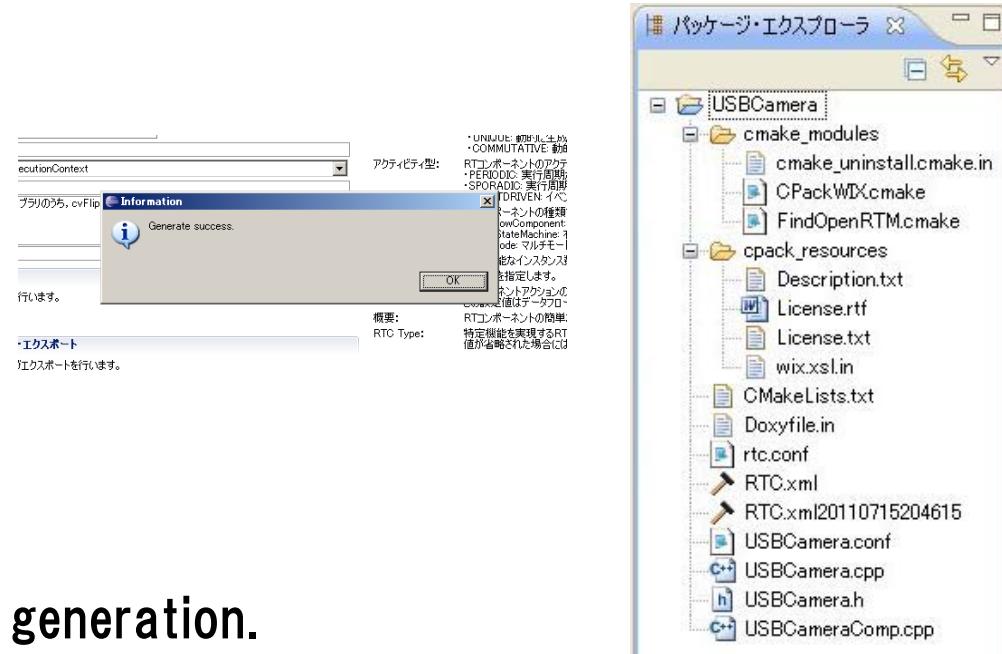
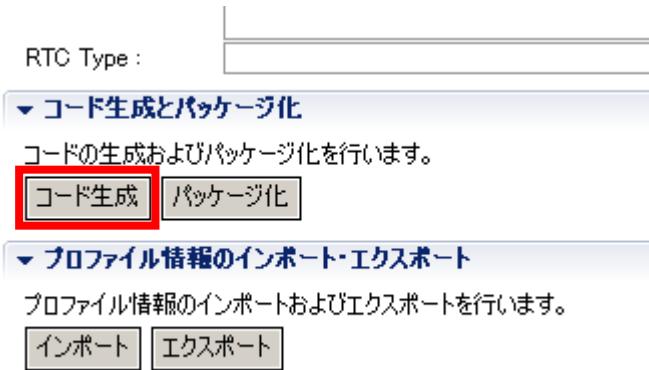


■ Reusing existing RTCProfiles.

- Existing RTCProfiles can be exported and imported.
- Imported RTCProfiles can be used to generate code.
- Files can be imported and exported in XML and YAML formats.

Code generation

■ Generate the code template



■ The perspective changes after generation.



If the code is not displayed, click "Refresh."

C++ RTC → CDT
Java RTC → JDT
Python RTC → PyDev

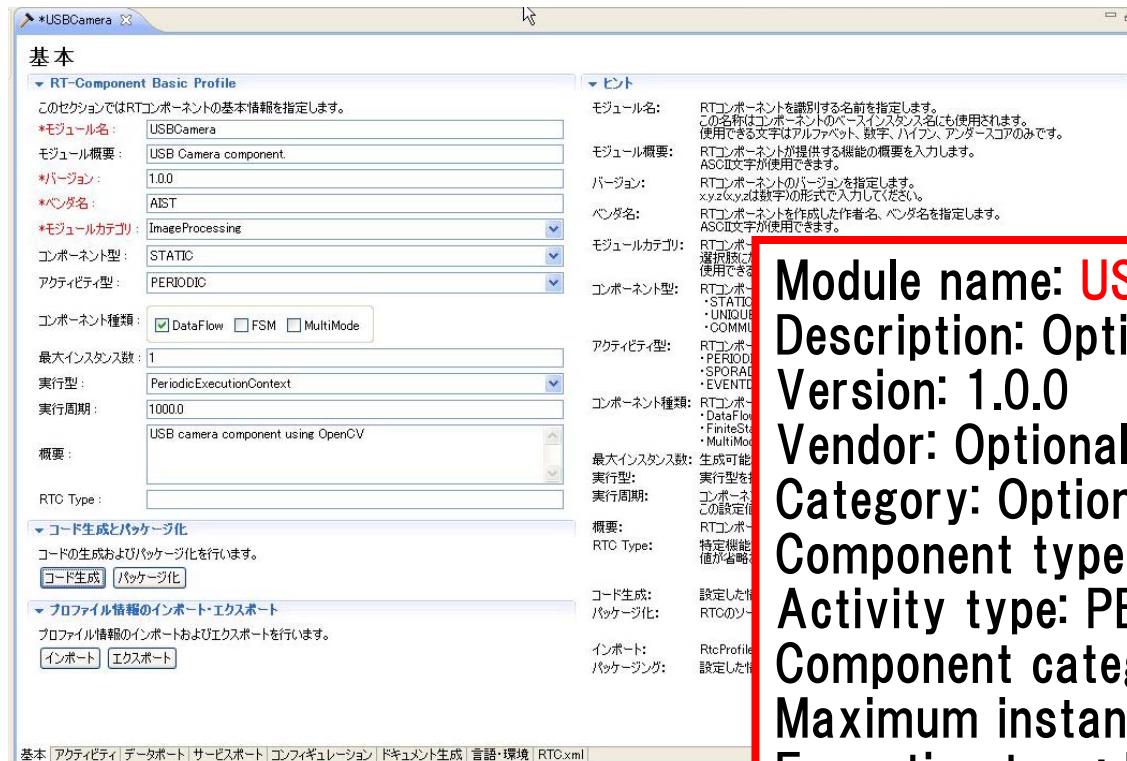
RTC Profile Editor



Tab	Explanation
Basic profile	Enter the RTC's basic profile and information. Generate code, import/export profiles and manage packaging.
Activity profile	Select the activities the RTC will support.
Data port profile	Manage the RTC's data ports.
Service port profile	Manage the RTC's service ports and the service interfaces they support.
Configuration	Manage the configuration parameters and sets that can be edited by the RTC's users, and system configuration parameters.
Documentation	Edit the documentation to add to the generated code.
Language, environment	Select the language to generate, set the environment.
RTC.xml	Displays the current RTCProfile in XML format.

Basic profile

■ RTコンポーネントの名称など、基本的な情報を設定



Module name: USBCamera
Description: Optional (USB Camera component)
Version: 1.0.0
Vendor: Optional (AIST)
Category: Optional (ImageProcessing)
Component type: STATIC
Activity type: PERIODIC
Component category: DataFlow
Maximum instances: 1
Execution type: PeriodicExecutionContext
Execution rate: 1000.0

※ Values in red must be provided.

※ Explanations are given on the right.

Activity profile

The activities to be implemented in the RTC.

アクティビティ

このセクションでは使用するアクションコールバックを指定します。

コンポーネントの初期化と終了処理に関するアクション	ビント
onInitialize	onFinalize
実行コンテキストの起動と停止に関するアクション	onStartup onShutdown
onStartup	onDeactivated
alive状態でのコンポーネントアクション	onActivated onDeactivated
onActivated	onAborting
onError	onReset
Dataflow型コンポーネントのアクション	onExecute onStateUpdate onRateChanged
onExecute	onRateChanged
FSM型コンポーネントのアクション	onAction onModeChanged
onAction	onExecute
Mode型コンポーネントのアクション	onModeChanged
onModeChanged	onRateChanged onAction onModeChanged

初期化処理です。コンポーネントライフサイクル開始時に一度だけ呼ばれます。常に有効。
終了処理です。コンポーネントライフサイクルの終了時に一度だけ呼ばれます。
ExecutionContextが実行を開始するとき1度だけ呼ばれます。
非アクティブ状態からアクティブ化されるとき1度だけ呼ばれます。
アクティブ状態から非アクティブ化されるとき1度だけ呼ばれます。
ERROR状態に入る前に1度だけ呼ばれます。
ERROR状態からリカバリー非アクティブ状態に移行するときに1度だけ呼ばれます。
アクティビティ状態時に割り貼りで呼ばれます。
onExecuteの後毎回呼ばれます。
ExecutionContextのrateが変更されたとき呼ばれます。
対応する状態に応じた動作を実行するために呼ばれます。
モードが変更された時に呼ばれます。

Documentation

このセクションでは各アクションの概要を説明するドキュメントを記述します。
上級のアクションを選択すると、それとのドキュメントを記述できます。

アクティビティ名: **onInitialize** ON OFF

動作概要: コンポーネント自身の各種初期化処理

事前条件: なし

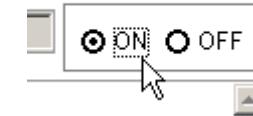
事後条件: コンポーネントの初期化処理が正常に完了している

基本 アクティビティ データポート サービスポート コンフィギュレーション ドキュメント生成 言語・環境 RTC.xml Mapping ID USB PnP RTC-CANopen

①Select the activity to edit.



②Turn it on or off.



Check the following:
onActivated
onDeactivated
onExecute

※The currently-selected activity is displayed in red.

※Enabled activities are highlighted in blue.

※All activities may have execution, pre- and post-condition documentation attached.

Data port profile

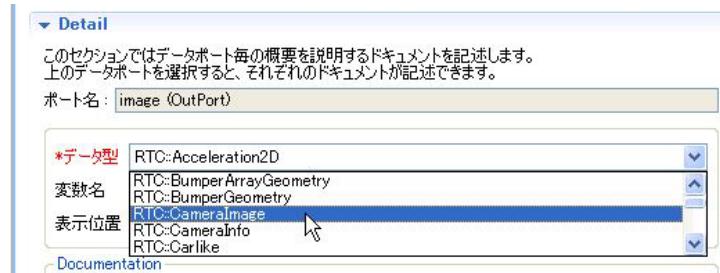
■ Data ports to add to the RTC.



① Select “Add” next to the type of port to add.



② Select the port's properties.



- ✖ The available data types are specified in IDL files, which must be set in the settings screen.
- ✖ The data types provided with OpenRTM-aist (in RTM_Root]rtm/idl) can be used by default.
- ✖ Documentation can be added to the ports.

Data port profile

✖ Port information is displayed in the build view.



● OutPort

Port name: **image**

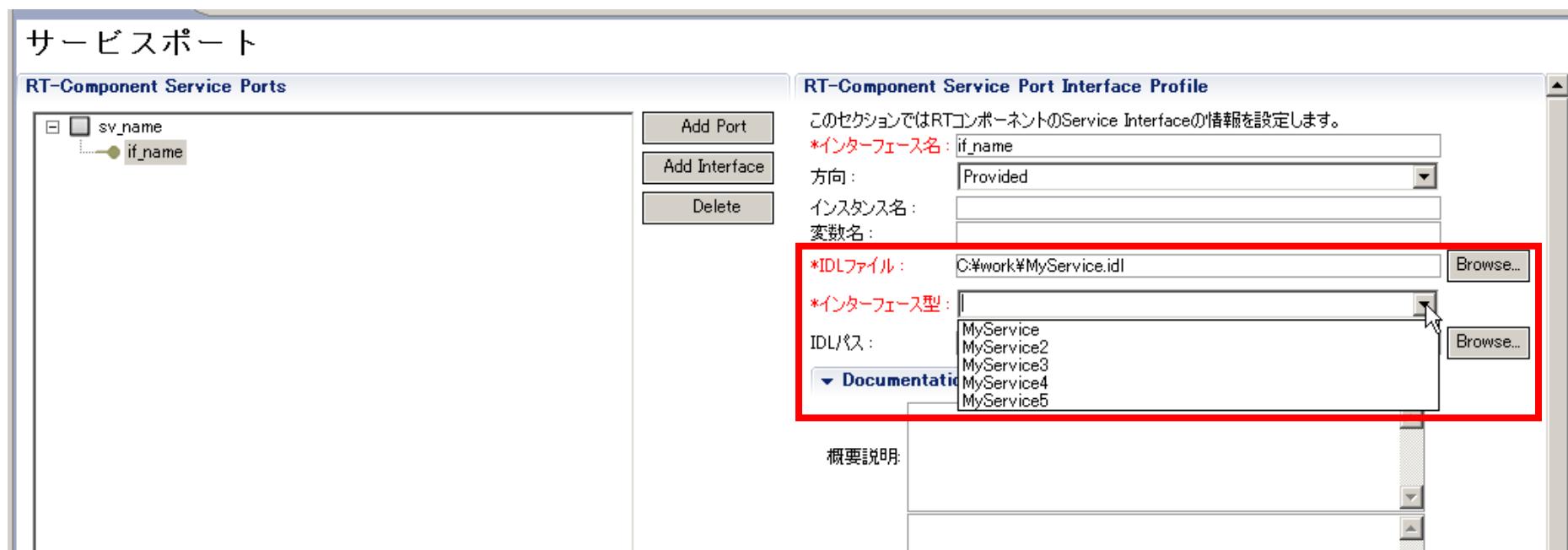
Data type: **RTC::Cameralmage**

Variable name: **image**

Display position: right

Service port profile

■ Service ports to add to the RTC.



- Service port interface settings
 - Specifying an IDL file displays the interfaces in that file.

This sample does not have
any service ports.

Configuration profile

■ Manage the configuration parameters of the RTC.

コンフィギュレーション・パラメータ

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

*名称	deviceNumber
Add	
Delete	

Detail

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

パラメータ名	deviceNumber
*データ型	int
*デフォルト値	0
変数名:	
単位:	
制約条件:	
Widget:	text
Step:	

Config. Param: RTコンポーネント用に利用可能なパラメーターは、コンフィギュレーション・パラメータ名にはアルファベット、数字、ハイフン、アンダースコアが含まれます。データ型: コンフィギュレーション・パラメータ名は名前にはアルファベット、数字、ハイフン、アンダースコアが含まれます。デフォルト値: コンフィギュレーション・パラメータの初期値です。変数名: 実際の変数名を指定します。単位: 単位を指定します。制約条件: 制約条件を指定します。Widget: ワイド: Step: 設定用

① Click the “Add” button and enter the parameter information.

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

*名称	conf_name0
Add	
Delete	

② Enter the variable name, etc.

Name: **deviceNumber**
 Data type: **int**
 Default value: **0**
 Variable: **deviceNumber**
 Constraints:
 Widget: **text**

※ Available data types include short, int, long, float, double, and string. Direct entry is also possible.

※ Constraints and widget information are used by RTSysEditor.

Setting constraints and widgets

■ About constraints

- Can be set on data ports and configuration parameters
- Checking **must** be performed by the developer.
 - Constraints do not mean the middleware enforces them.

■ Entering constraints

- No constraints: Blank
- Direct: Use the value as-is
 - e.g. 100
- Range:<, >, <=, >=
 - e.g. 0<=x<=100
- Enumeration: (值1, 値2, ...)
 - e.g. (val0, val1, val2)
- Array: Value 1, Value 2, ...
 - e.g. val0, val1, val2
- Key-value: { Key 0:Value 0, Key 1:Value 1, ... }
 - e.g. { key0:val0, key1:val1}

■ Widget

- text (Default)
 - slider
 - For numerical values with a range
 - Set the step size with “step”
 - spinbox
 - For numerical values with a range
 - Set the step size with “step”
 - radio button
 - For enumerations
- ※ When the widget and constraints do not match, text is used.

Language and environment

- Set the language to generate and environment settings

言語・環境

このセクションでは使用する言語を指定します

C++
 Python
 Java
 Ruby

Use old build environment.

▼ ヒント

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

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

このセクションでは依存するライブラリや使用するOSなどを指定します

Version	OS

Add Delete

詳細情報

OS Version

Add Delete

CPU

Add Delete

このチェックボックスをONにすると、
旧バージョンと同様なコード(Cmake
を利用しない形式)を生成

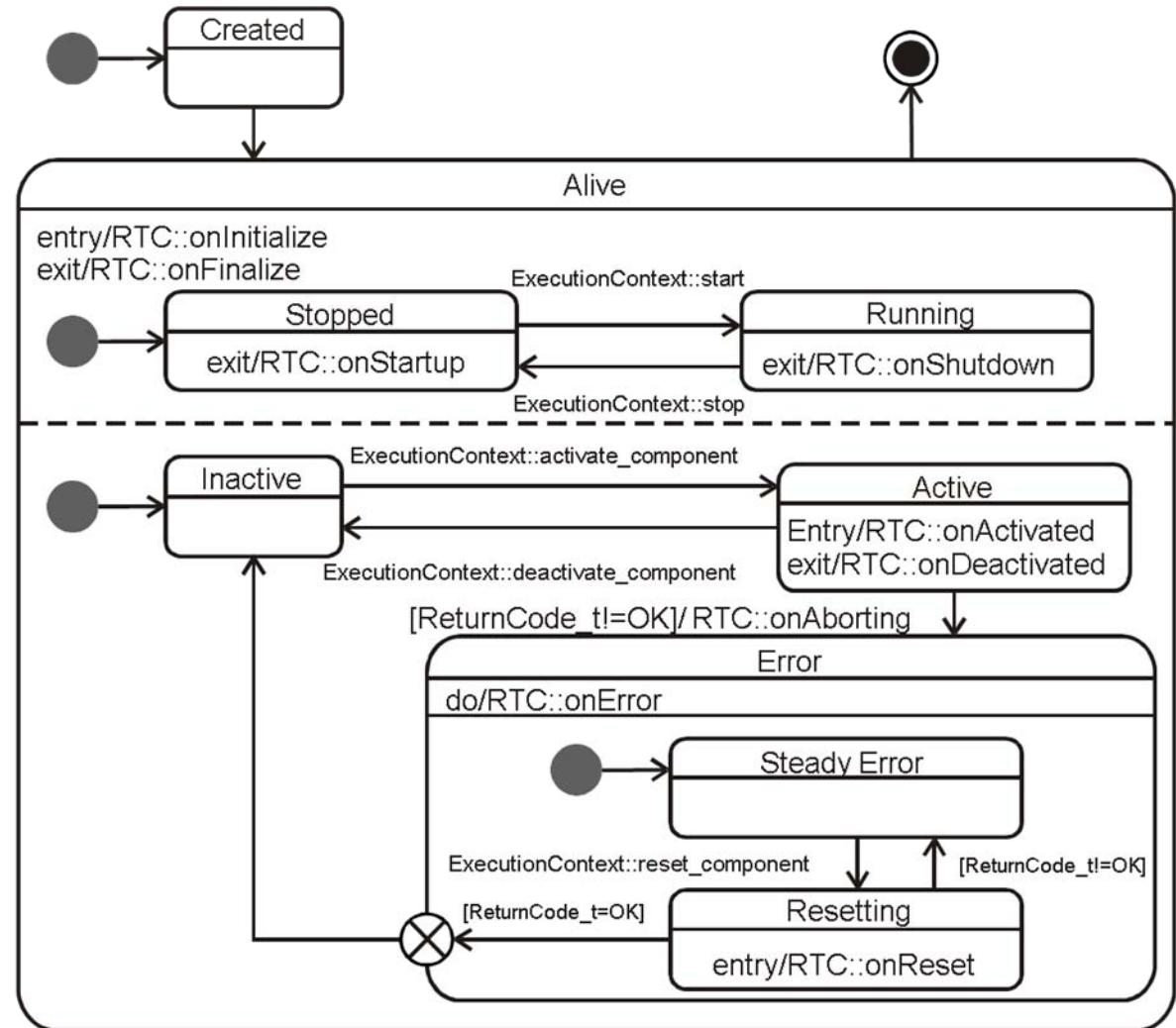
Select C++

Generating the RT-Component



Implementing an RT Component

- RTC states
 - Created
 - Alive
 - Inactive
 - Active
 - Error
 - Finalised



RTC lifecycle (UML state chart)

Implementing an RT Component



■ Available call-back functions

関数名	概要
onInitialize	Called once when the lifecycle starts.
onActivated	Called once when the component is activated.
onDeactivated	Called once when the component is deactivated.
onExecute	Called regularly while the component is active.
onStateUpdate	Called after onExecute.
onAborting	Called once when changing to the Error state.
onError	Called regularly while in the Error state.
onReset	Called once when leaving the Error state.
onShutdown	Called once when the EC shuts down.
onStartup	Called once when the EC starts.
onFinalize	Called once when the lifecycle ends.

Implementing the component

■ Make a component from an existing program.

```
int main (int argc, char** argv) {  
    // カメラからの画像をキャプチャするクラスの  
    // インスタンスを生成  
    ds_Camera *cam;  
    cam = new ds_Camera();  
  
    // キャプチャクラスのオブジェクトの初期化  
    cam->initialize();  
  
    while(1) {  
        // カメラからの画像キャプチャ処理  
        cam->capture();  
  
        // 画像を表示  
        cvShowImage("Capture", cam->getImage());  
        cvWaitKey(2);  
    };  
  
    // キャプチャクラスのオブジェクトの終了処理  
    cam->finalize();  
  
    // キャプチャクラスのオブジェクトの破棄  
    delete cam;  
  
    return 0;  
}
```

RTC::onInitialize()

RTC::onActivated()

RTC::onExecute()

RTC::onDeactivated()

USBCamera RTC source

Implementing the component

■ To make this an RTC:

```
RTC::ReturnCode_t USBCamera::onInitialize() {  
    // Set OutPort buffer  
    addOutPort("image", m_image0ut);  
    // Bind variables and configuration variable  
    bindParameter("deviceNumber", m_deviceNumber, "0");  
  
    //Create an instance of the camera class  
    cam = new ds_Camera();  
    return RTC::RTC_OK;  
}
```

```
RTC::ReturnCode_t USBCamera::onFinalize() {  
    // Delete the camera object  
    delete cam;  
    return RTC::RTC_OK;  
}
```

Implementing the component

- To make this an RTC:

```
RTC::ReturnCode_t
USBCamera::onActivated(RTC::UniqueId ec_id) {
    // Initialise the camera object
    if(cam->initialize())
        return RTC::RTC_OK;
    return RTC::RTC_ERROR;
}
```

```
RTC::ReturnCode_t
USBCamera::onDeactivated(RTC::UniqueId ec_id) {
    // Shut down the camera object
    cam->finalize();
    return RTC::RTC_OK;
}
```

Implementing the component



```
RTC::ReturnCode_t USBCamera::onExecute(RTC::UniqueId ec_id) {
    // Capture an image from the camera
    if (cam->capture() < 0)
        return RTC::RTC_OK;

    // Get the image size
    int len = cam->getImageSize();
    CvSize size = cam->getSize();

    // Set up the output data
    m_image.pixels.length(len);
    m_image.width  = size.width;
    m_image.height = size.height;

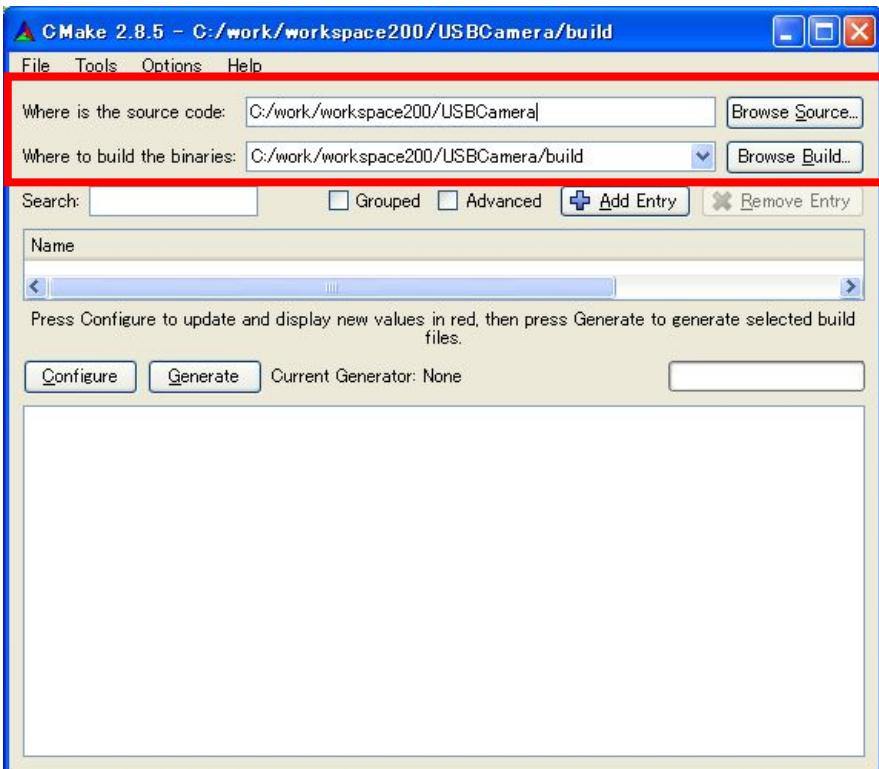
    // Add the image data to the output
    memcpy((void *)&(m_image.pixels[0]), cam->getImageData(), len);

    // Write the output
    m_imageOut.write();

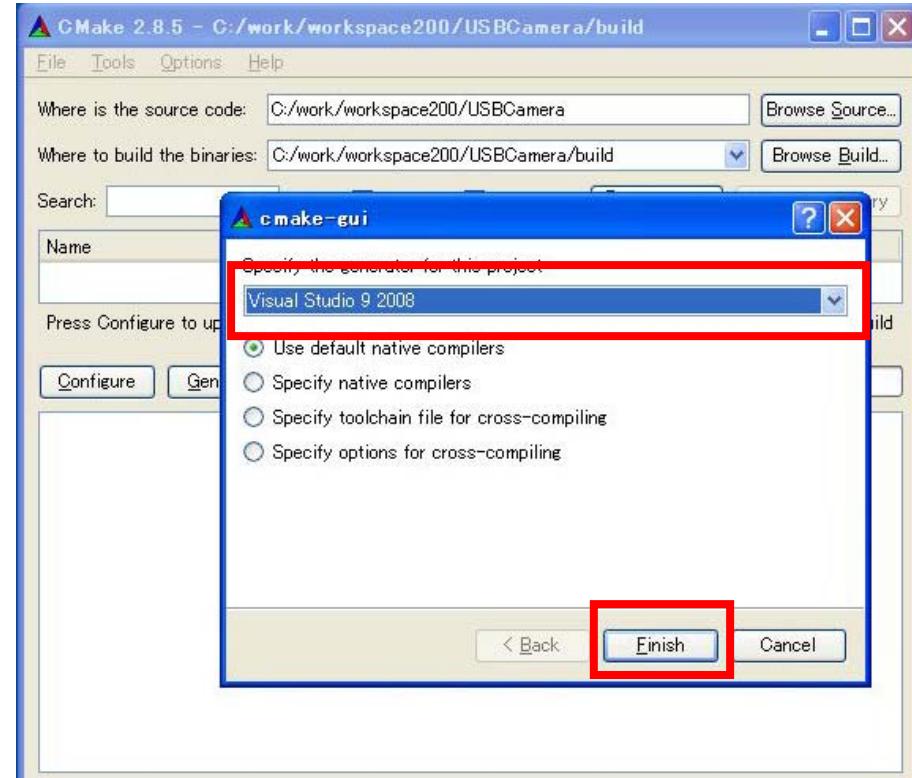
    return RTC::RTC_OK;
}
```

Compiling (Windows, CMake)

① Start the CMake GUI and specify the source and binary directories.



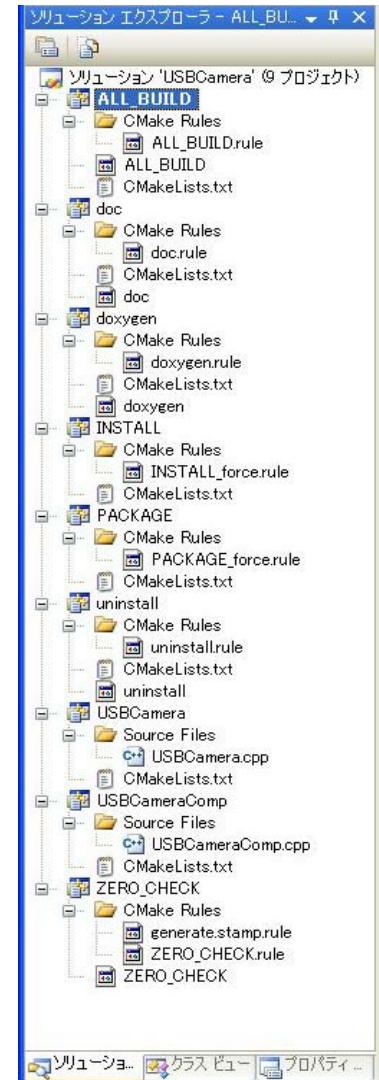
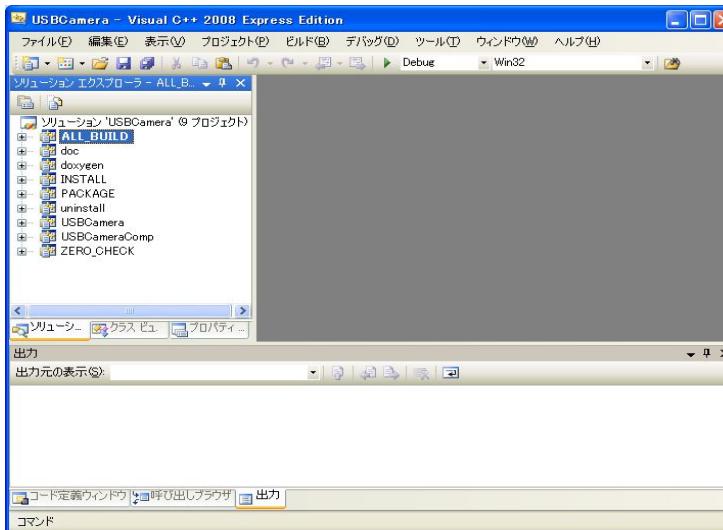
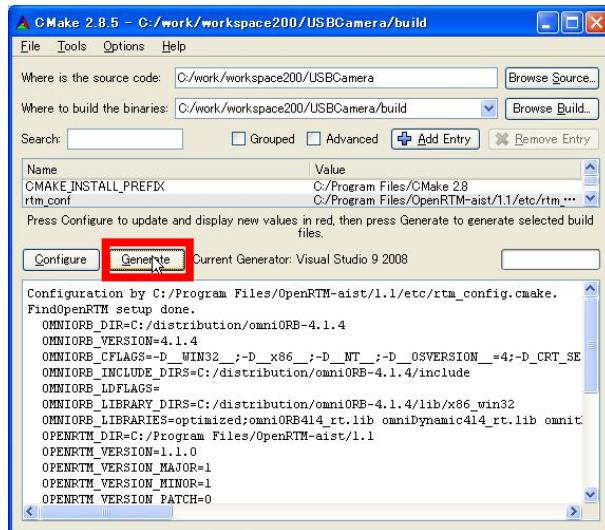
② Press “Configure” and select the platform to build for.



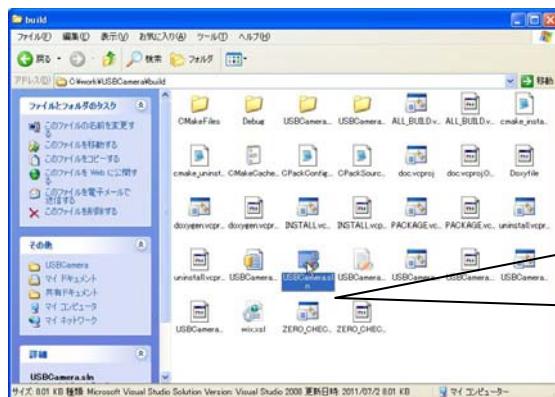
✖ Specify separate directories for the binary and source directories.

Compiling (Windows, CMake)

③ Click “Generate” once configuration has completed.

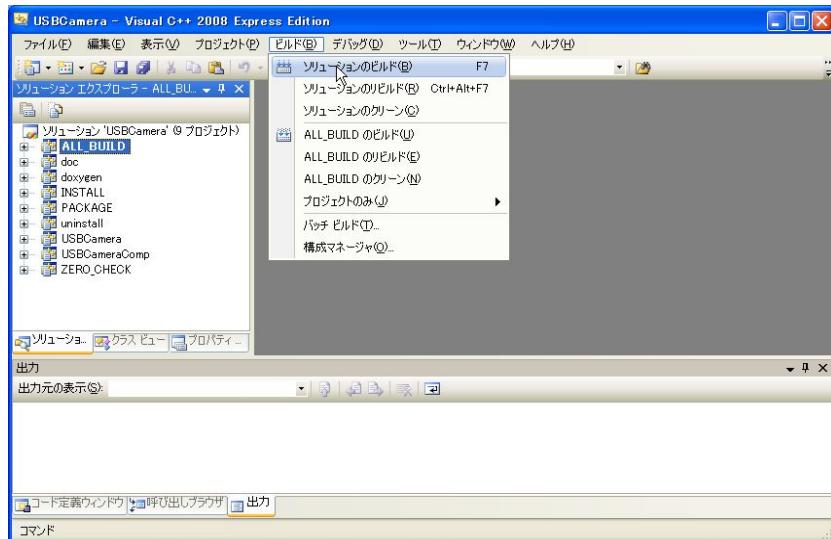


④ Open the solution file in the binary directory.

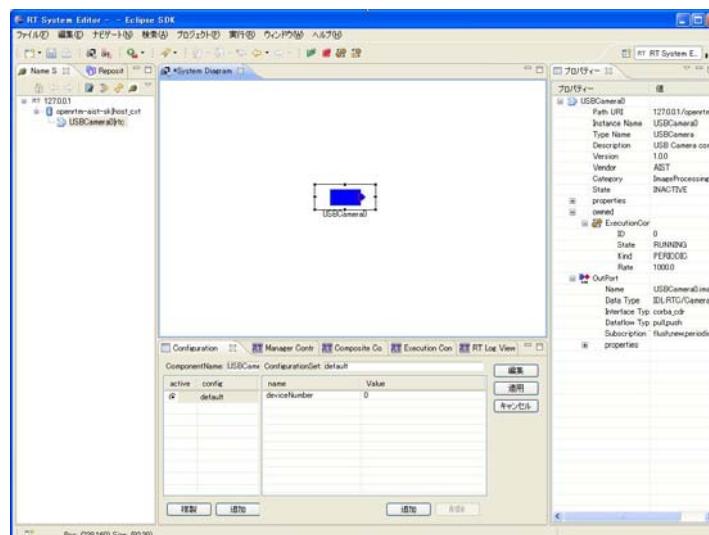
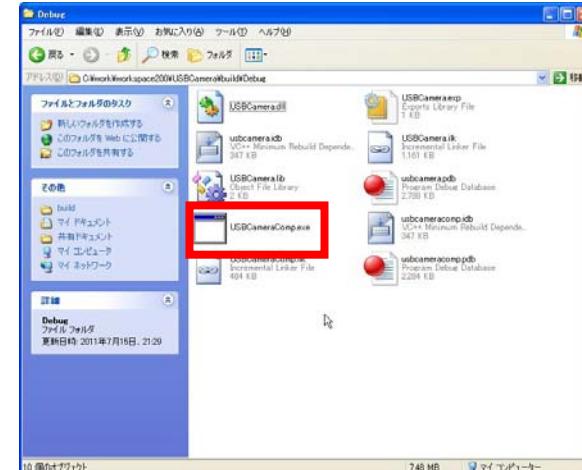


Compiling (Windows, CMake)

⑤Build the solution



⑥In the “Debug” directory of the binary directory, start the component.



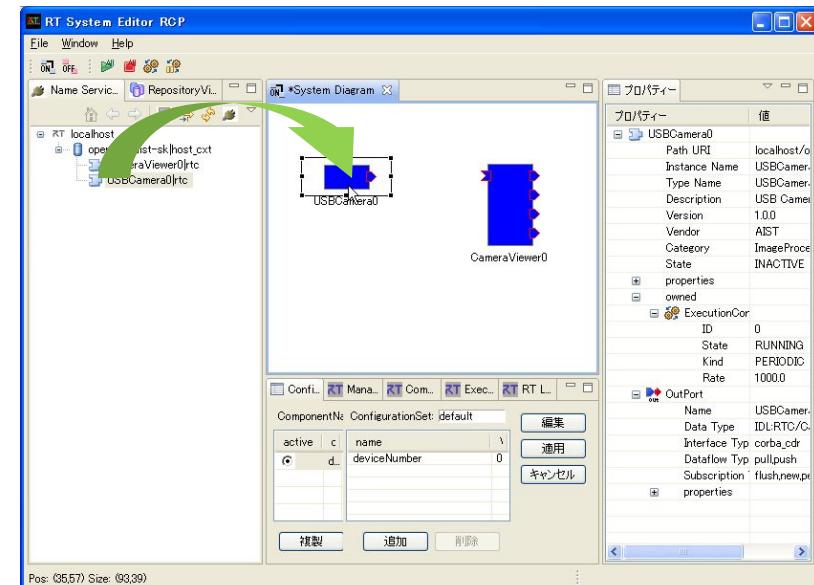
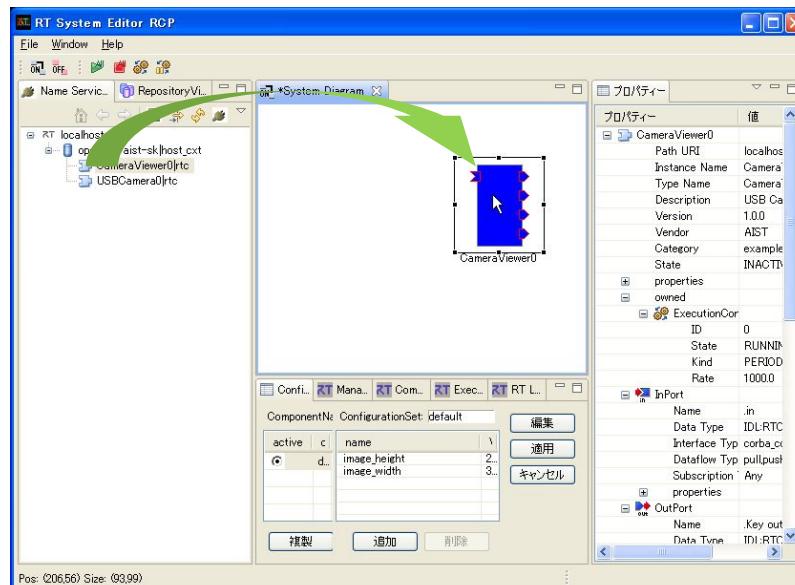
Check operation

1. Start the CameraViewer

- [Start menu]→[All programs]→[OpenRTM-aist 1.1]→[C++]→[components]
→[opencv_rtcs]→[CameraViewerComp.exe]

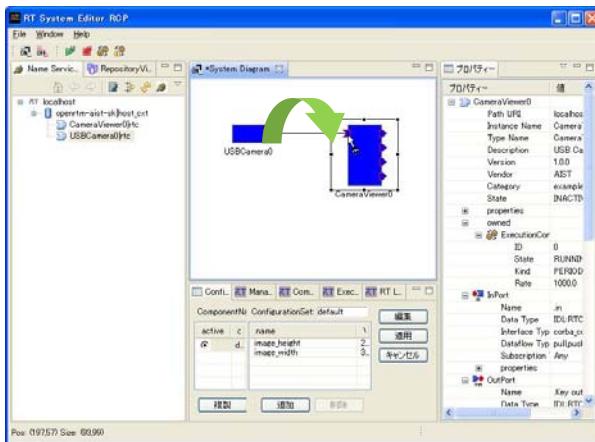
2. Connect the components.

Drag and drop the components into the system editor.

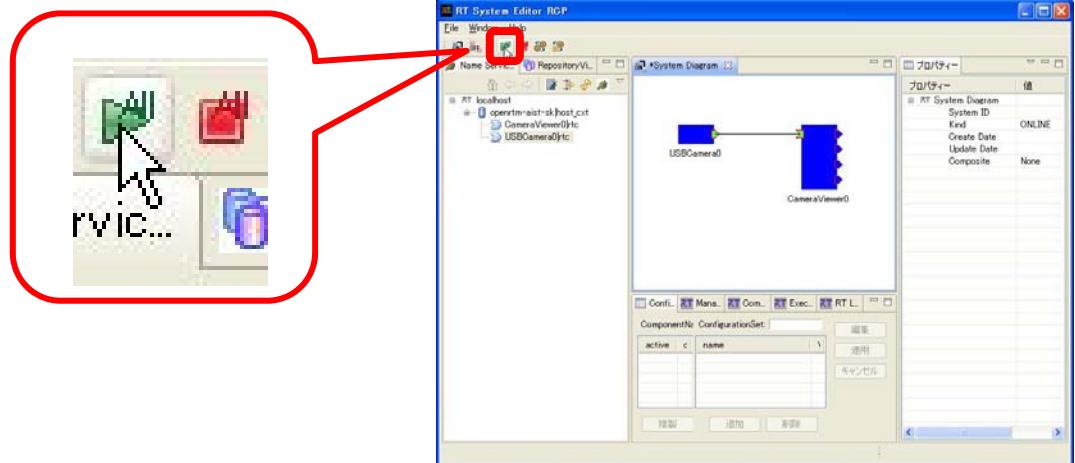


Check operation

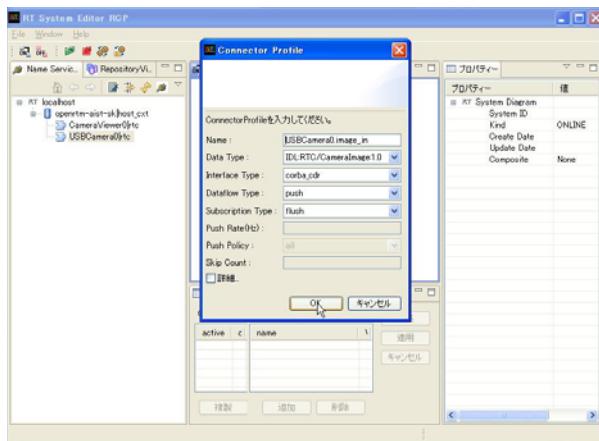
3. Connect the ports.



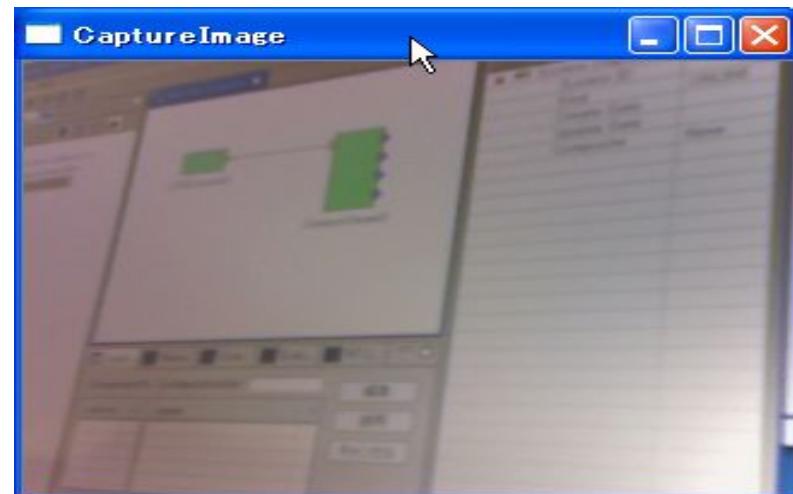
5. Activate the components.



4. Use the default profile.



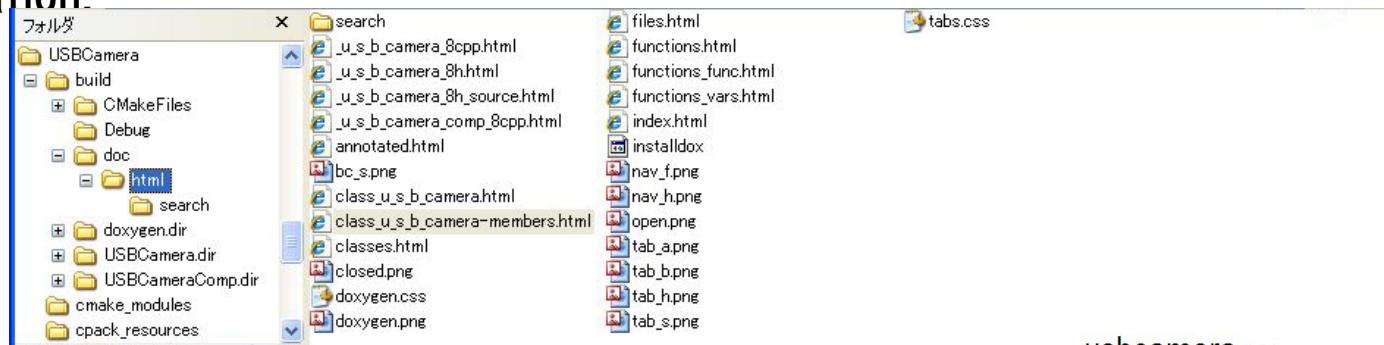
6. The camera image will be displayed.



Document generation (Windows, CMake)



※The doc/html/ directory in the binary directory contains Doxygen-generated documentation.



Example:

usbcamera 1.0.0

クラス USBCamera

USB Camera component. [詳細]

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

すべてのメンバ一覧

Public メソッド

USBCamera (RTC::Manager *manager) constructor

~USBCamera () destructor

virtual RTC::ReturnCode_t onInitialize ()

virtual RTC::ReturnCode_t onActivated (RTC::UniqueId ec_id)

virtual RTC::ReturnCode_t onDeactivated (RTC::UniqueId ec_id)

virtual RTC::ReturnCode_t onExecute (RTC::UniqueId ec_id)

Protected 変数

int m_deviceNumber

CameraImage m_image

OutPort<CameraImage> m_imageOut

このクラスの説明は以下のファイルから生成されました:

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

usbcamera 1.0.0

メインページ クラス ファイル

ファイル一覧

C:/work/workspace199/USBCamera/USBCamera.h

説明を見る。

```
00001 // -*- C++ -*-
00015 #ifndef USBCAMERA_H
00016 #define USBCAMERA_H
00017
00018 #include <rtm/Manager.h>
00019 #include <rtm/DataFlowComponentBase.h>
00020 #include <rtm/CorbaPort.h>
00021 #include <rtm/DataInPort.h>
00022 #include <rtm/DataOutPort.h>
00023 #include <rmw/idl/BasicDataTypeSkel.h>
00024 #include <rmw/idl/ExtendedDataTypesSkel.h>
00025 #include <rmw/idl/InterfaceDataTypesSkel.h>
00026
00027 // Service implementation headers
00028 // <rtc-template block="service_impl_h">
00029
00030 // </rtc-template>
00031
00032 // Service Consumer stub headers
00033 // <rtc-template block="consumer_stub_h">
00034
00035 // </rtc-template>
00036
00037 using namespace RTC;
00038
00050 class USBCamera : public RTC::DataFlowComponentBase
00051 {
00052 public:
00053     USBCamera(RTC::Manager* manager);
00059
00063     ~USBCamera();
00064
00065 // <rtc-template block="public_attribute">
00066 }
```

Add some more components

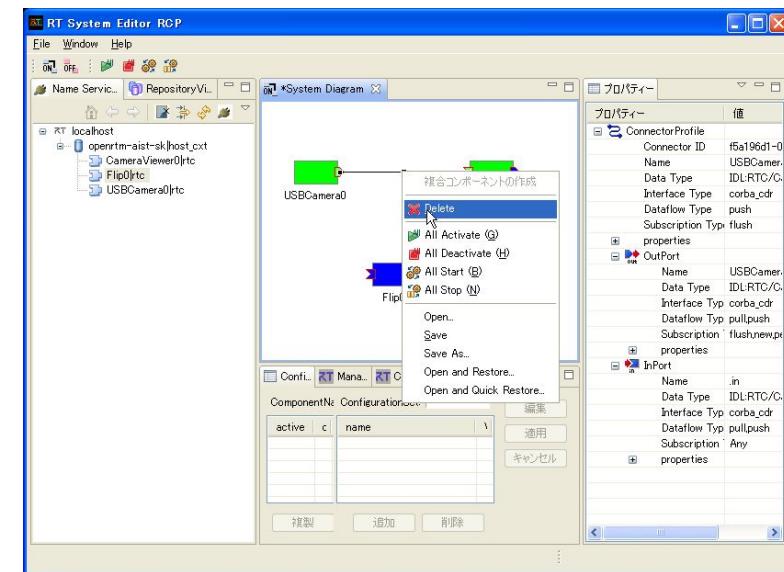
1. Start the Flip component

- [Start menu]→[All programs]→[OpenRTM-aist 1.1]→[C++]→[components]
→[opencv_rtcs]→[FlipComp.exe]

2. Add the Flip component to the system editor.

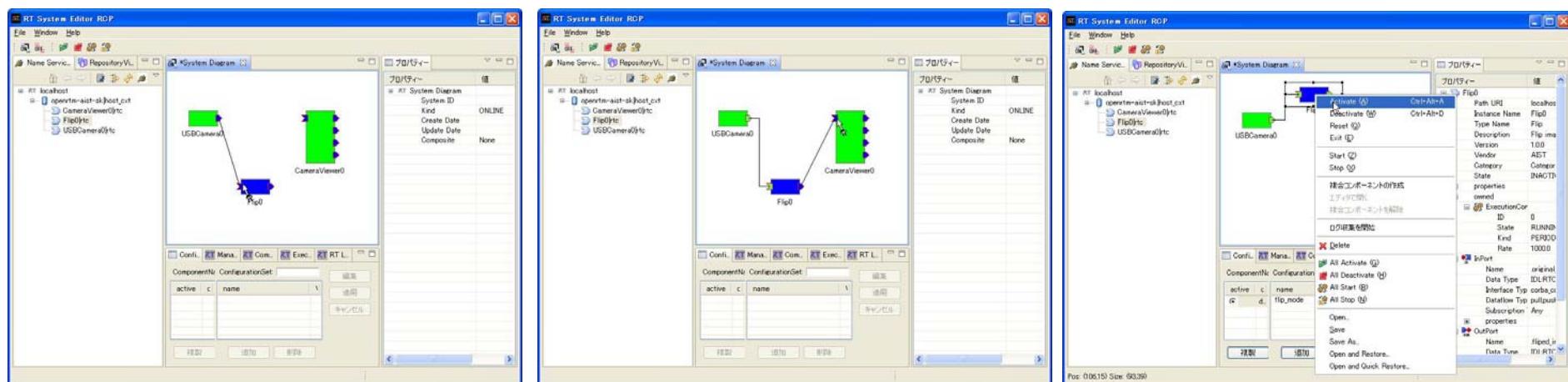
3. Disconnect the USBCamera and CameraViewer components.

- (1) Select the connection line.
- (2) Right-click on it.
- (3) Click “Delete.”



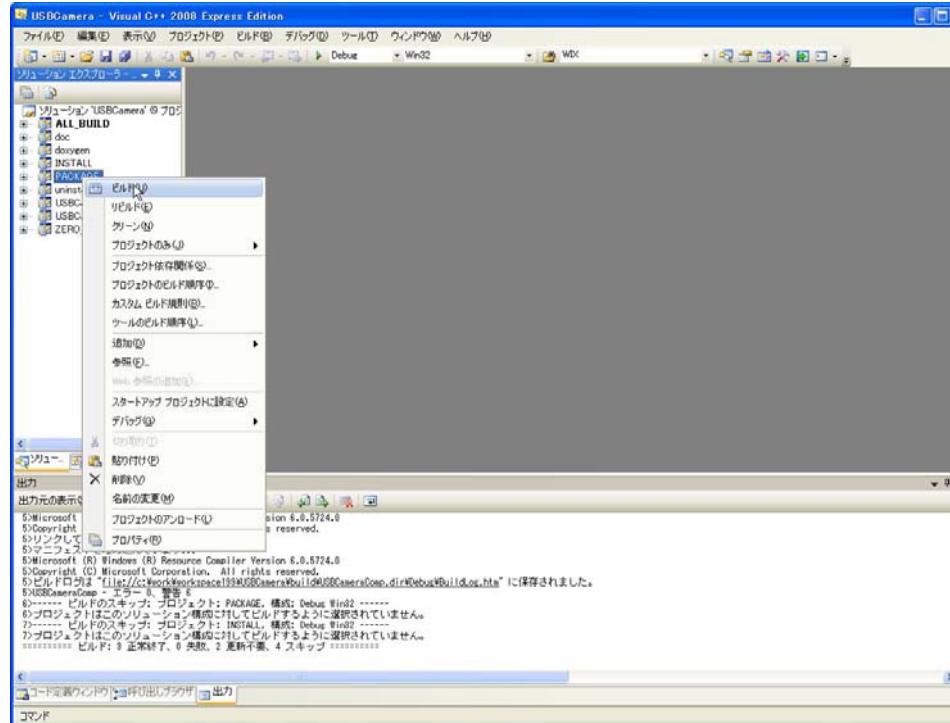
Add some more components

1. Connect USBCamera's OutPort to Flip's InPort.
 2. Connect Flip's OutPort to CameraViewer's InPort.
 3. Activate the Flip component.
- (1) Right-click on Flip.
 (2) Select “Activate” from the context menu.



Distributable package generation

- Build the “PACKAGE” target in the solution.



- An MSI installer is generated in the binary directory.
 - The component is installed into:
C:/Program Files/OpenRTM-aist/1.1/components/<Language>/<Package name>

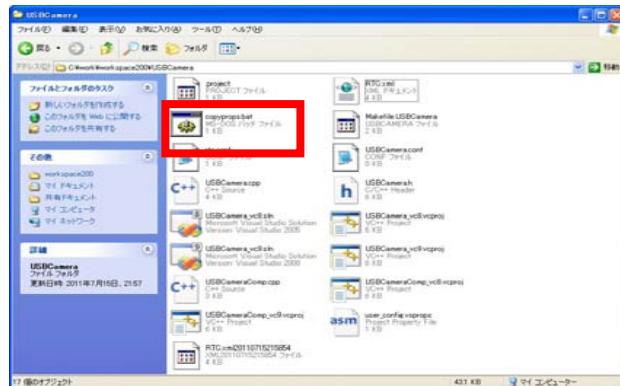
Supplement



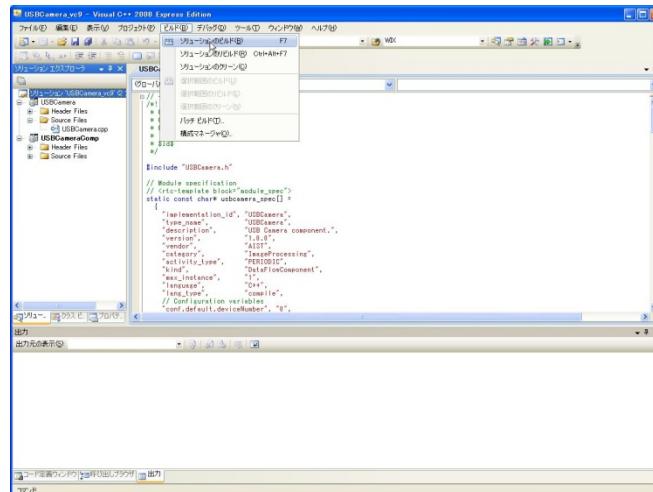
Compiling on Windows without CMake



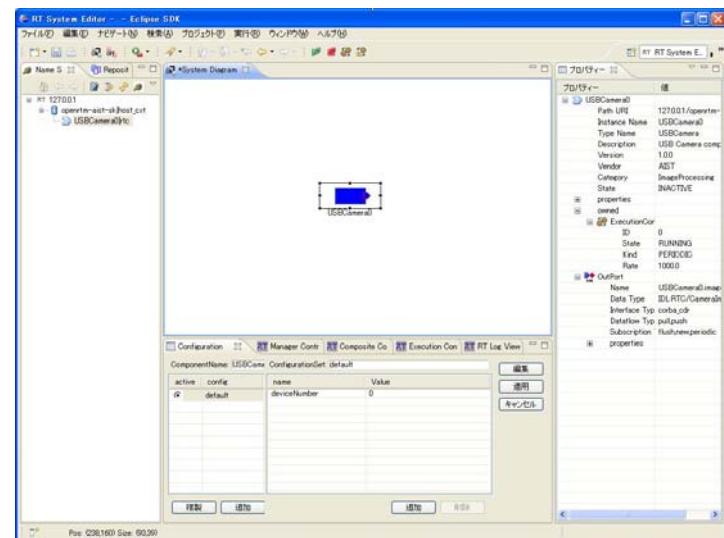
① Execute “copyprops.bat” in the generated code directory to copy the properties files.



② Build with Visual Studio

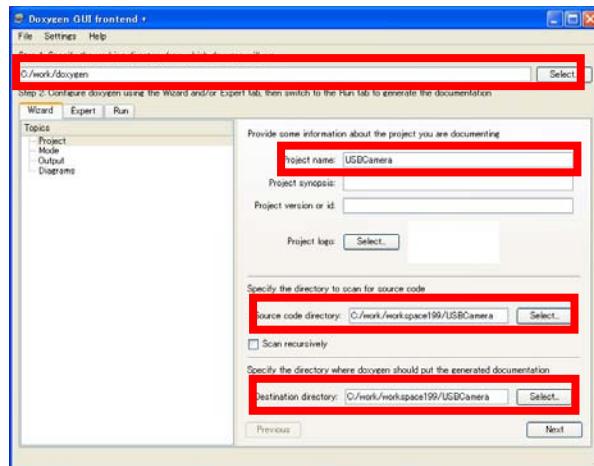


③ In the USBCameraComp/Debug directory, execute USBCameraComp.exe

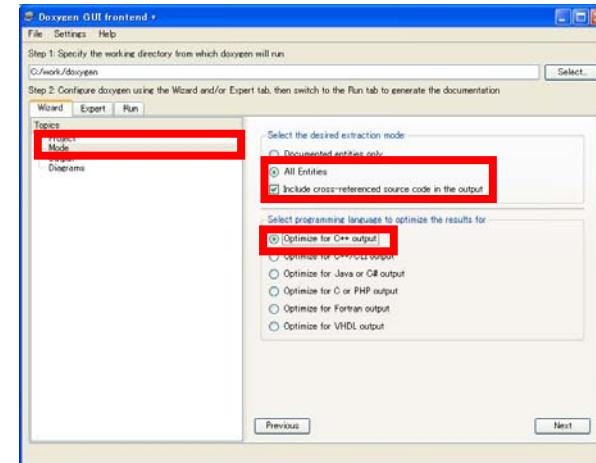


Document generation without CMake

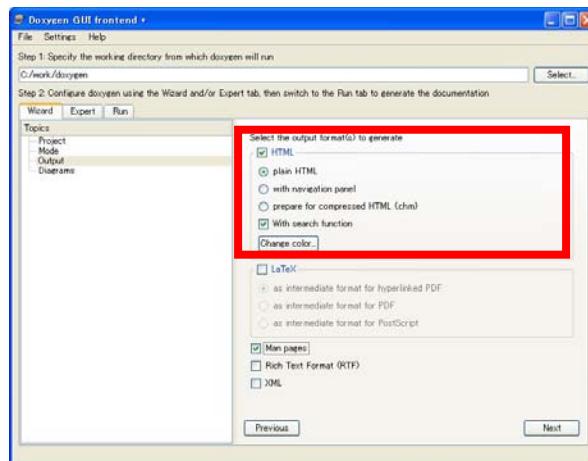
① Start the Doxygen GUI tool.
Set the directories and project name.



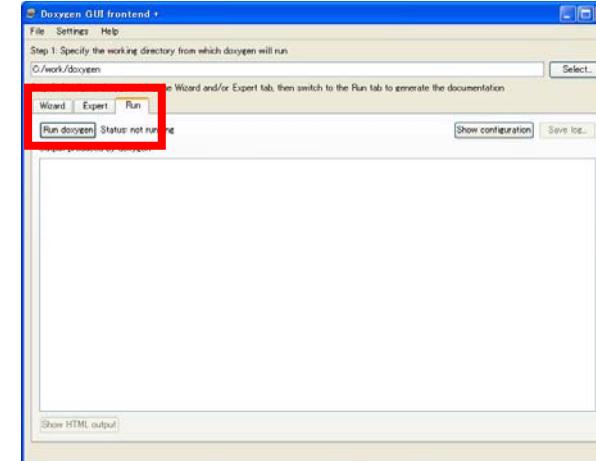
② Set the output and language settings in the “Mode” section.



③ In the “Output” section, select HTML.



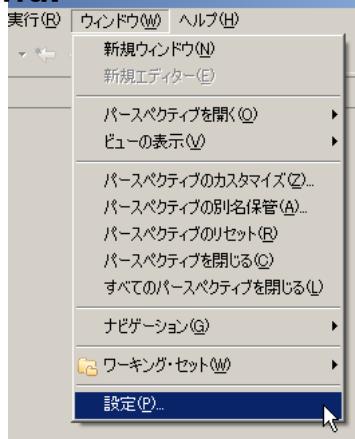
③ Click “Run Doxygen” in the Run tab.



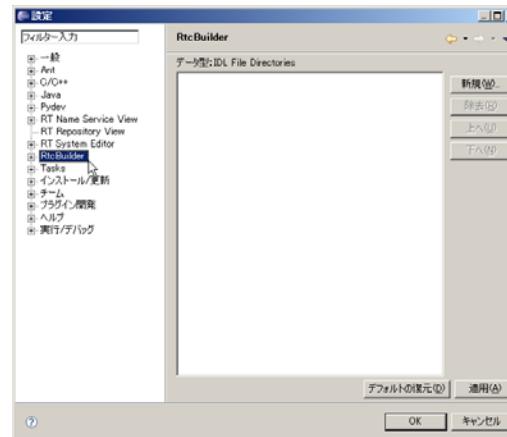
Various settings

- Specify data types available for DataPorts by specifying the directories containing the IDL files.

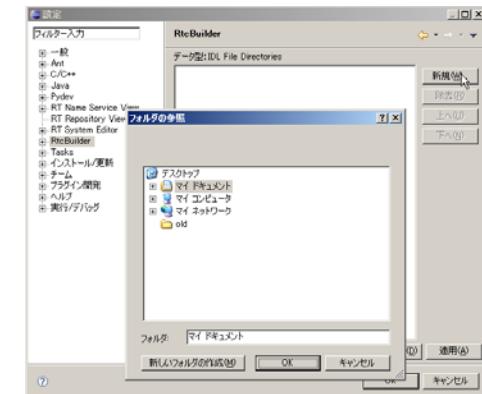
①Select “Settings” from the “Window” menu.



②Select “RtcBuilder.”



③Click the “Append” button and select the directory.



※Only necessary when using your own IDL files. The OpenRTM-aist default data types are set automatically.

- Default data types are available at [RTM_Root]rtm/idl
 →BasicDataType.idl, ExtendedDataTypes.idl, etc.
 →By default, [RTM_Root]=C:/Program Files/OpenRTM-aist/1.1/

SICE 2011

RT-Middleware Tutorial

