

青梅商工会議所主催 RTM講習会

日時:2011年7月25日(月) 10:30~17:30

場所:産業技術総合研究所 中央第2 本部情報棟1F ネットワーク会議室



10:30- 11:15	第1部:RTミドルウェアの概略紹介
	担当:神徳 徹雄(産業技術総合研究所)
	概要:RTミドルウェア, RTコンポーネントの概要説明
11:15- 12:00	第2部:RTミドルウェアの概略, 導入方法の紹介
	担当:栗原真二(産業技術総合研究所)
	概要:サンプルシステムを用いた概略紹介. RTミドルウェアの導入方法について紹介
13:00- 14:00	第3部:RTミドルウェアを用いたシステム構築方法の紹介
	担当:坂本武志(株式会社グローバルアシスト)
	概要:簡単なサンプルの動作(実習1)基本的な仕組み, 機能の紹介
14:15- 15:15	第4部:RTコンポーネントの作成方法の紹介
	担当:坂本武志(株式会社グローバルアシスト), 栗原真二(産業技術総合研究所)
	概要:サンプルコンポーネントの作成(実習2)RTCの設計方法の紹介
15:30- 16:00	第5部:OpenRTM-aistコマンドラインツール rtshellの利用方法
	担当:Geoffrey Biggs(産業技術総合研究所)
	概要:rtshellの紹介と, システムの実行. ログのとりかた, ログの再生方法の紹介
16:15- 17:30	第6部:RTミドルウェアの便利な機能
	担当:片見 剛人(富士ソフト株式会社)
	概要:ネットワークで接続されたシステムの紹介(実習3)ホームページプロジェクト登録の例. 質疑応答

第4部 RTコンポーネントの作成方法の紹介

株式会社 グローバルアシスト 坂本 武志
産業技術総合研究所 栗原 眞二



RTCBuilderについて

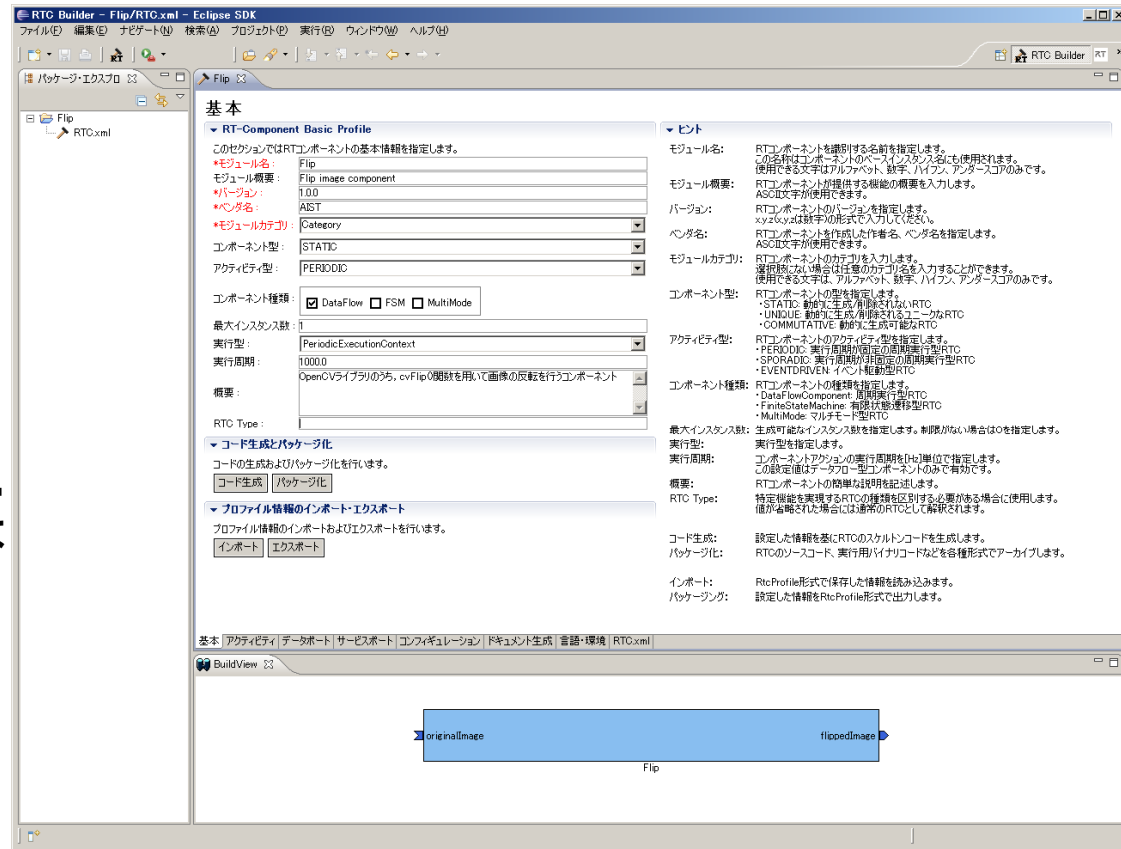


■ RTCBuilderとは？

- コンポーネントのプロファイル情報を入力し、ソースコード等の雛形を生成するツール
- 開発言語用プラグインを追加することにより、各言語向けRTCの雛形を生成することが可能

- C++
- Java
- Python

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



画面構成

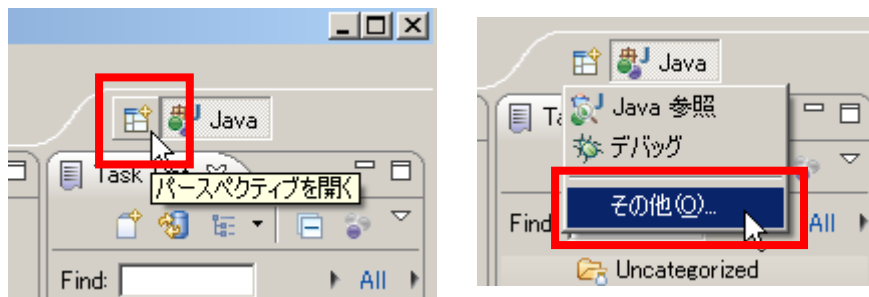
The screenshot shows the RTC Builder application window with the following components and annotations:

- Package Explorer:** Located on the left, showing the project structure. A purple dashed box highlights it, with a callout box containing the text "パッケージ・エクスプローラ".
- Basic Profile Editor:** The main central area, titled "基本" (Basic). It contains fields for:
 - Module Name: Flip
 - Module Summary: Flip image component
 - Version: 1.0.0
 - Vendor Name: AIST
 - Module Category: Category
 - Component Type: STATIC
 - Activity Type: PERIODIC
 - Component Kind: DataFlow FSM MultiMode
 - Maximum Instance Count: 1A blue dashed box highlights this area, with a callout box containing the text "RTCプロファイルエディタ".
- Hints:** A green dashed box highlights the right-hand side of the interface, which contains detailed instructions for each field. A green callout box with the text "ヒント" (Hint) points to this section.
- Code Generation and Packaging:** A section at the bottom of the basic editor with buttons for "コード生成" (Code Generation) and "パッケージ化" (Packaging).
- Profile Information Import/Export:** A section at the bottom of the basic editor with buttons for "インポート" (Import) and "エクスポート" (Export).
- Build View:** Located at the bottom of the window, showing a diagram of the "Flip" component. It consists of two blue boxes labeled "originalImage" and "flippedImage" connected by a blue arrow. A red dashed box highlights this area, with a callout box containing the text "ビルドビュー".

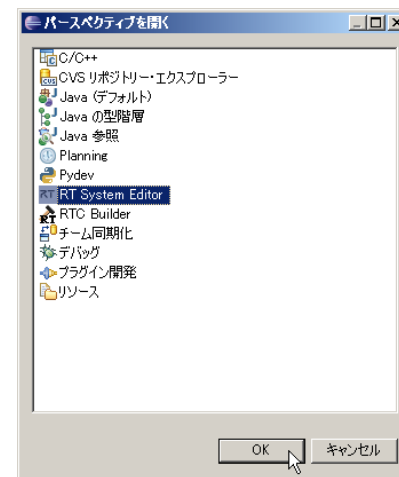
- 以下から「USBCamera.zip」をダウンロードします。
<http://www.openrtm.org/openrtm/ja/node/1677#document>
- USBCamera.zipをC:¥に展開します。
 - ※ USBCameraが、スペースを含むパスに展開された場合、VC++でのビルド時にエラーが発生します。
“C:¥”でなくても、スペースを含まないところであれば構いません。

■ パースペクティブの切り替え

- ① 画面右上の「パースペクティブを開く」を選択し、一覧から「その他」を選択



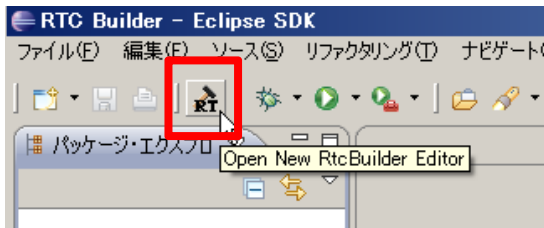
- ② 一覧画面から対象ツールを選択



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

プロジェクト作成/エディタ起動

① ツールバー内のアイコンをクリック



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

※任意の場所にプロジェクトを作成したい場合

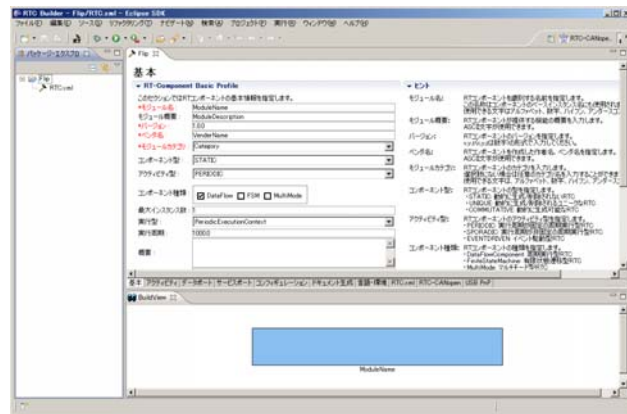
- ②にて「デフォルト・ロケーションの使用」チェックボックスを外す
- 「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名: USBCamera

② 「プロジェクト名」欄に入力し、「終了」

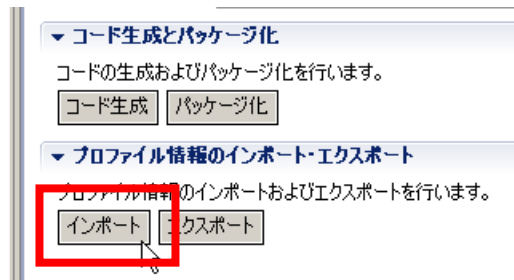


③ 指定した名称のプロジェクトを生成

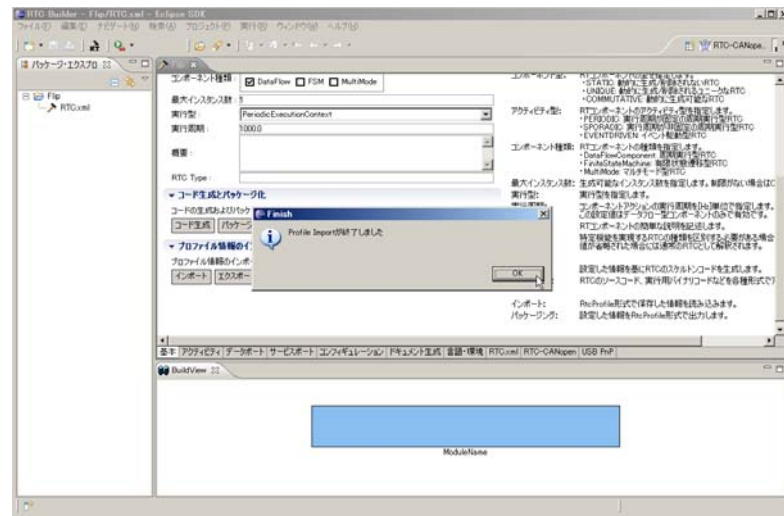


プロファイル インポート

①「基本」タブ下部の「インポート」ボタンをクリック



②【インポート】画面にて対象ファイルを選択



■ 作成済みのRTコンポーネント情報を再利用

- 「エクスポート」機能を利用して出力したファイルの読み込みが可能
- コード生成時に作成されるRtcProfileの情報を読み込み可能
- XML形式, YAML形式での入出力が可能

■ コード生成

RTC Type :

▼ コード生成とパッケージ化

コードの生成およびパッケージ化を行います。

コード生成

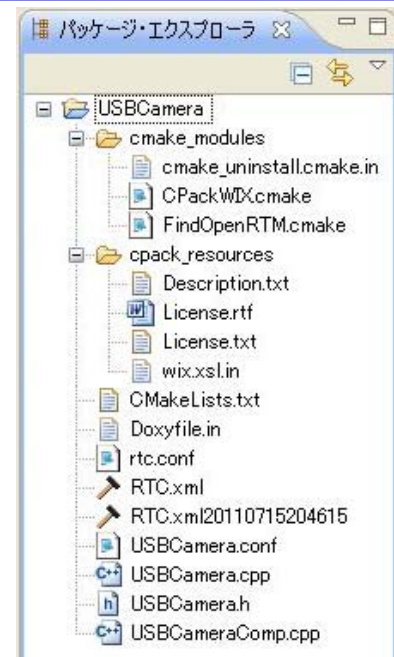
パッケージ化

▼ プロファイル情報のインポート・エクスポート

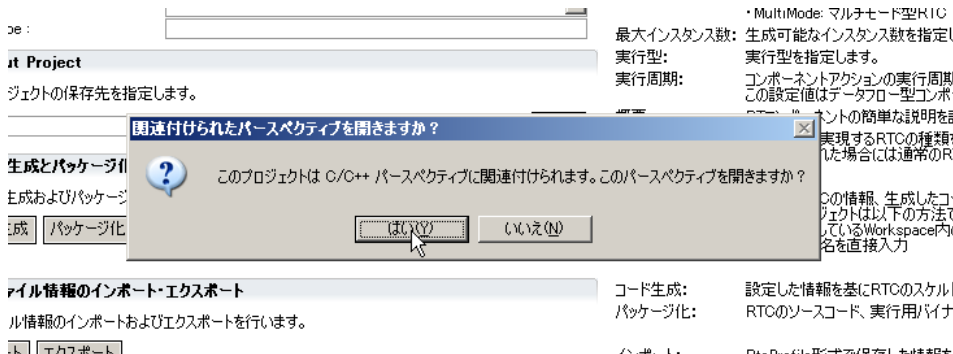
プロファイル情報のインポートおよびエクスポートを行います。

インポート

エクスポート



■ コード生成実行後、パースペクティブを自動切替

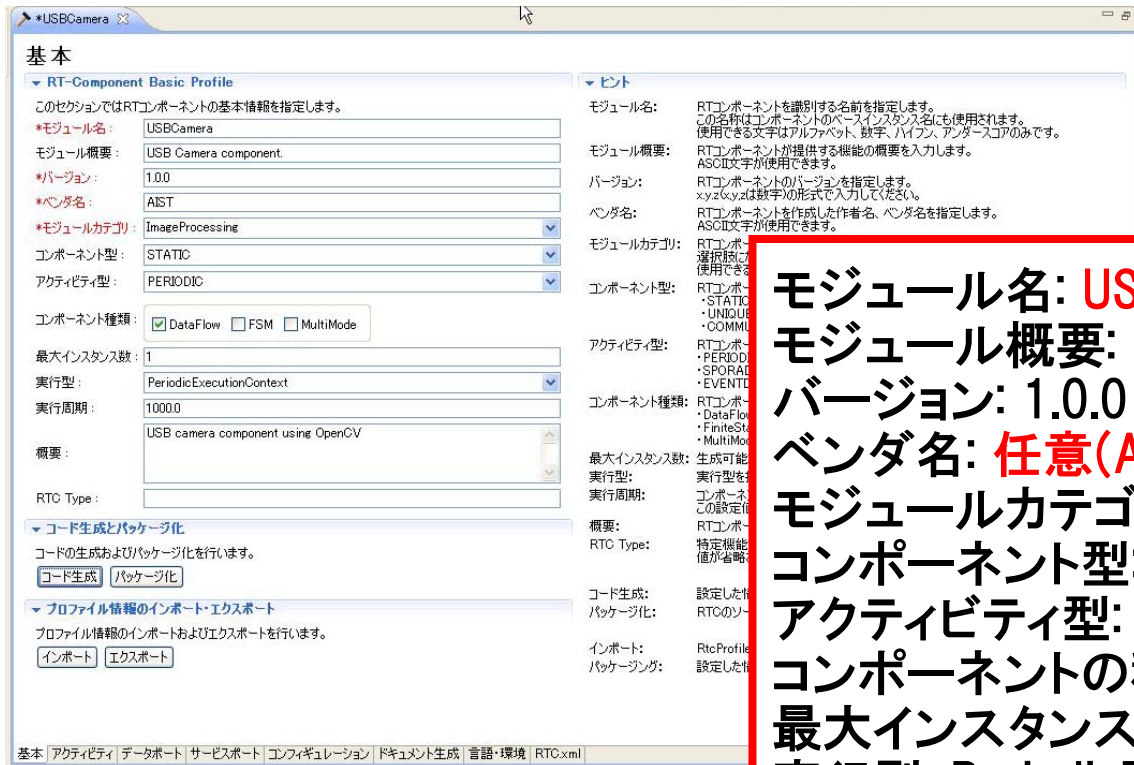


※生成コードが表示されない場合には、「リフレッシュ」を実行

- C++版RTC → CDT
- Java版RTC → JDT
- (デフォルトインストール済み)
- Python版 → PyDev

画面要素名	説明
基本プロフィール	RTコンポーネントのプロファイル情報など、コンポーネントの基本情報を設定。 コード生成, インポート/エクスポート, パッケージング処理を実行
アクティビティ・プロファイル	RTコンポーネントがサポートしているアクティビティ情報を設定
データポート・プロファイル	RTコンポーネントに付属するデータポートに関する情報を設定
サービスポート・プロファイル	RTコンポーネントに付属するサービスポートおよび各サービスポートに付属するサービスインターフェースに関する情報を設定
コンフィギュレーション	RTコンポーネントに設定するユーザ定義のコンフィギュレーション・パラメータセット情報およびシステムのコンフィギュレーション情報を設定
ドキュメント生成	生成したコードに追加する各種ドキュメント情報を設定
言語・環境	生成対象コードの選択やOSなどの実行環境に関する情報を設定
RTC.xml	設定した情報を基に生成したRTC仕様(RtcProfile)を表示

RTコンポーネントの名称など、基本的な情報を設定



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

- ❌ エディタ内の項目名が赤字の要素は必須入力項目
- ❌ 画面右側は各入力項目に関する説明

■ 生成対象RTCで実装予定のアクティビティを設定

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。

onInitialize	onFinalize	
onStartup	onShutdown	
onActivated	onDeactivated	onAborting
onError	onReset	
onExecute	onStateUpdate	onRateChanged
onAction		
onModeChanged		

▼ ヒント

onInitialize	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びます。常に有効。
onFinalize	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びます。
onStartup	ExecutionContextが実行を開始するとき一度だけ呼びます。
onShutdown	ExecutionContextが実行を停止するとき一度だけ呼びます。
onActivated	非アクティブ状態からアクティブ化されたとき一度だけ呼びます。
onDeactivated	アクティブ状態から非アクティブ化されたとき一度だけ呼びます。
onAborting	ERROR状態に入る前に一度だけ呼びます。
onError	ERROR状態に在る間周期的に呼びます。
onReset	ERROR状態からリセットされ非アクティブ状態に移行するとき一度だけ呼びます。
onExecute	アクティブ状態時に周期的に呼びます。
onStateUpdate	onExecuteの毎回の呼びます。
onRateChanged	ExecutionContextのrateが変更されたとき呼びます。
onAction	対応する状態に応じた動作を実行するために呼びます。
onModeChanged	モードが変更された時に呼びます。

動作概要: アクティビティの概要説明を記述します。
事前条件: アクティビティを実行する前に成立すべき事前条件を記述します。
事後条件: アクティビティを実行した後に成立すべき事後条件を記述します。

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。
上段のアクションを選択すると、それぞれのドキュメントを記述できます。

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

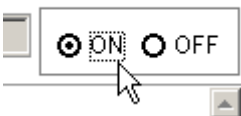
事後条件: コンポーネントの初期化処理が正常に完了している

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

① 設定対象のアクティビティを選択



② 使用/未使用を設定



以下をチェック:
onActivated
onDeactivated
onExecute

- ※現在選択中のアクティビティは、一覧画面にて赤字で表示
- ※使用(ON)が選択されているアクティビティは、一覧画面にて背景を水色で表示
- ※各アクティビティには、「動作概要」「事前条件」「事後条件」を記述可能
→記述した各種コメントは、生成コード内にDoxygen形式で追加される

■ 生成対象RTCに付加するDataPortの情報を設定

データポート

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

*ポート名 (InPort)	*ポート名 (OutPort)
	image

Detail

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

ポート名: image (OutPort)

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

Documentation

概要説明: Capture images data from the camera

データ型: RTC:CameraImage
データ数:
意味:

- ① 該当種類の欄の「Add」ボタンをクリックし、ポートを追加後、直接入力で名称設定

ポートの情報を設定します。

- ② 設定する型情報を一覧から選択

Detail

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

ポート名: image (OutPort)

*データ型: RTC:Acceleration2D
変数名: RTC:BumperArrayGeometry
表示位置: RTC:CameraImage
RTC:CameraInfo
RTC:Carlike

Documentation

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

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

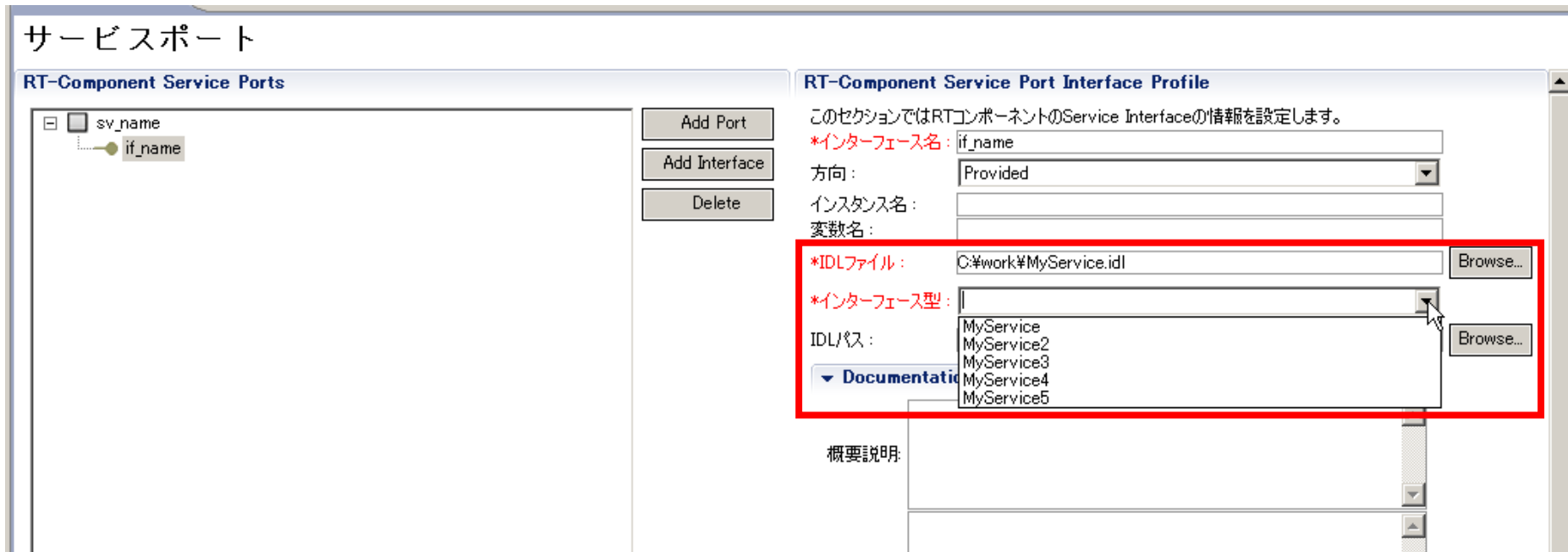
❌ 各ポートに対する説明記述を設定可能
→ 記述した各種コメントは、生成コード内にDoxygen形式で追加される

※Portの設定内容に応じて、下部のBuildViewの表示が変化



- OutPort
ポート名: **image**
データ型: **RTC::CameraImage**
変数名: **image**
表示位置: **right**

■ 生成対象RTCに付加するServicePortの情報を設定



■ サービスインターフェースの指定

- IDLファイルを指定すると, 定義されたインターフェース情報を表示

今回のサンプルでは未使用

■ 生成対象RTCで使用する設定情報を設定

コンフィギュレーション・パラメータ

①「Add」ボタンをクリックし、追加後、直接入力で名称設定

②詳細画面にて、型情報、変数名などを設定

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

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

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

■ 制約条件について

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

■ 制約の記述書式

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

■ Widget

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

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

■ 生成対象RTCを実装する言語，動作環境に関する情報を設定

言語・環境

▼ 言語

このセクションでは使用する言語を指定します

- C++
- Python
- Java
- Ruby

Use old build environment.

▼ 環境

このセクションでは依存するライブラリや使用するOSなどを指定します

Version	OS

Add
Delete

詳細情報

OS Version

Add
Delete

CPU

Add
Delete

▼ ヒント

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

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

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

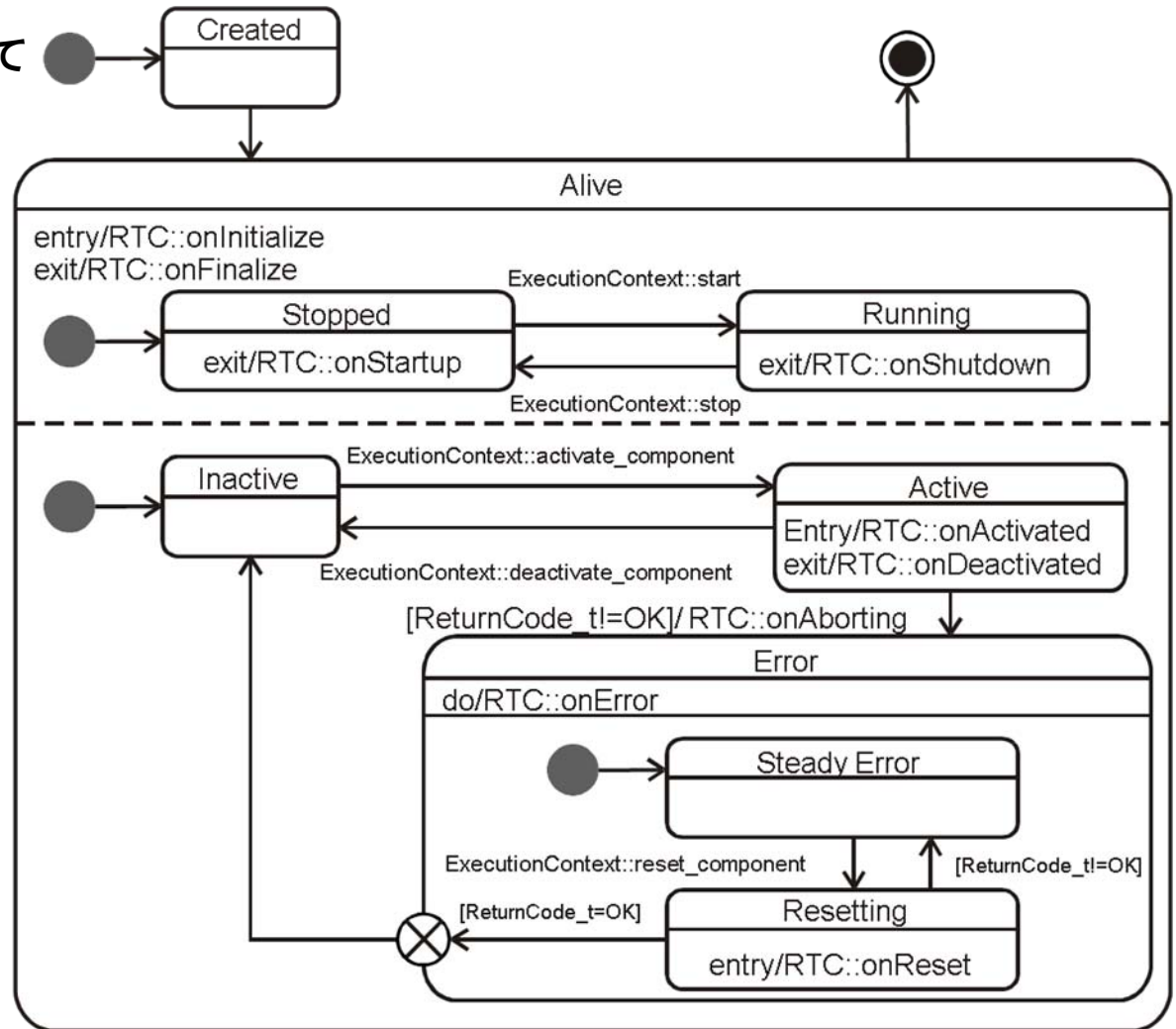
「C++」を選択

RTコンポーネントの作成



RTコンポーネントの実装(概要)

- RTコンポーネントの状態について
 - 生成状態(Created)
 - 活動状態(Alive)
 - 非アクティブ状態(Inactive)
 - アクティブ状態(Active)
 - エラー状態(Error)
 - 終了状態



RTCライフサイクル (UML ステートマシン図)

■ 予め決められた関数 (コールバック関数)について

関数名	概要
onInitialize	ライフサイクル初期化時に1度だけ呼ばれる。
onActivated	アクティブ化する際に1回呼ばれる。
onDeactivated	非アクティブ化する際に1回呼ばれる。
onExecute	アクティブ状態にあるとき周期的に呼ばれる。
onStateUpdate	onExecute の後に毎回呼ばれる。
onAborting	エラー状態に移行する際に1回呼ばれる。
onError	エラー状態にあるとき周期的に呼ばれる。
onReset	エラー状態から復帰する際に1回呼ばれる。
onShutdown	ECの駆動が停止する際に1回呼ばれる。
onStartup	ECの駆動が開始する際に1回呼ばれる。
onFinalize	ライフサイクル終了時に1度だけ呼ばれる。

■ 単体で動作確認済みのプログラムからRTコンポーネントを作成

```
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() に実装

RTC::onActivated() に実装

RTC::onExecute() に実装

RTC::onDeactivated() に実装

USBカメラコンポーネントのソースファイル

■ RTコンポーネントにすると

```
RTC::ReturnCode_t USBCamera::onInitialize() {  
    // Set OutPort buffer  
    addOutPort("image", m_imageOut);  
    // Bind variables and configuration variable  
    bindParameter("deviceNumber", m_deviceNumber, "0");  
  
    //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  
    cam = new ds_Camera();  
    return RTC::RTC_OK;  
}
```

```
RTC::ReturnCode_t USBCamera::onFinalize() {  
    // キャプチャクラスのオブジェクトの破棄  
    delete cam;  
    return RTC::RTC_OK;  
}
```

■ RTコンポーネントにすると

```
RTC::ReturnCode_t
USBCamera::onActivated(RTC::UniqueId ec_id) {
    // キャプチャクラスのオブジェクトの初期化
    if(cam->initialize())
        return RTC::RTC_OK;
    return RTC::RTC_ERROR;
}
```

```
RTC::ReturnCode_t
USBCamera::onDeactivated(RTC::UniqueId ec_id) {
    // キャプチャクラスのオブジェクトの終了処理
    cam->finalize();
    return RTC::RTC_OK;
}
```

```
RTC::ReturnCode_t USBCamera::onExecute(RTC::UniqueId ec_id) {
    // カメラからの画像キャプチャ処理
    if (cam->capture() < 0)
        return RTC::RTC_OK;

    // 画像サイズの取得
    int len = cam->getImageSize();
    CvSize size = cam->getSize();

    // アウトポート変数へ画像情報をセット
    m_image.pixels.length(len);
    m_image.width = size.width;
    m_image.height = size.height;

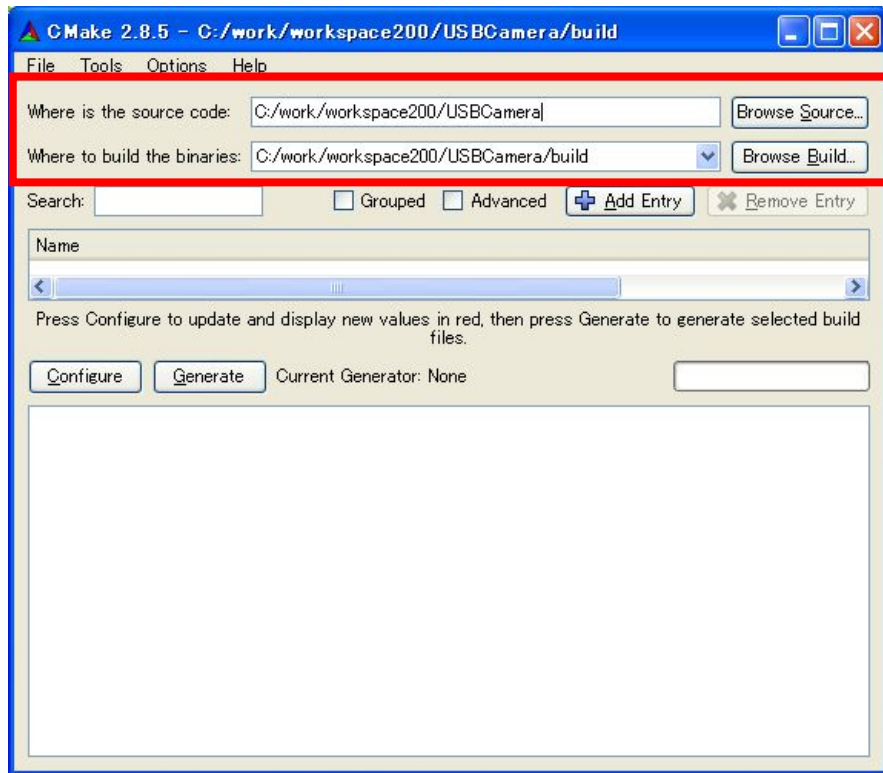
    // アウトポート変数へ画像データをセット
    memcpy((void *)&(m_image.pixels[0]), cam->getImageData(), len);

    // アウトポートからデータ出力
    m_imageOut.write();

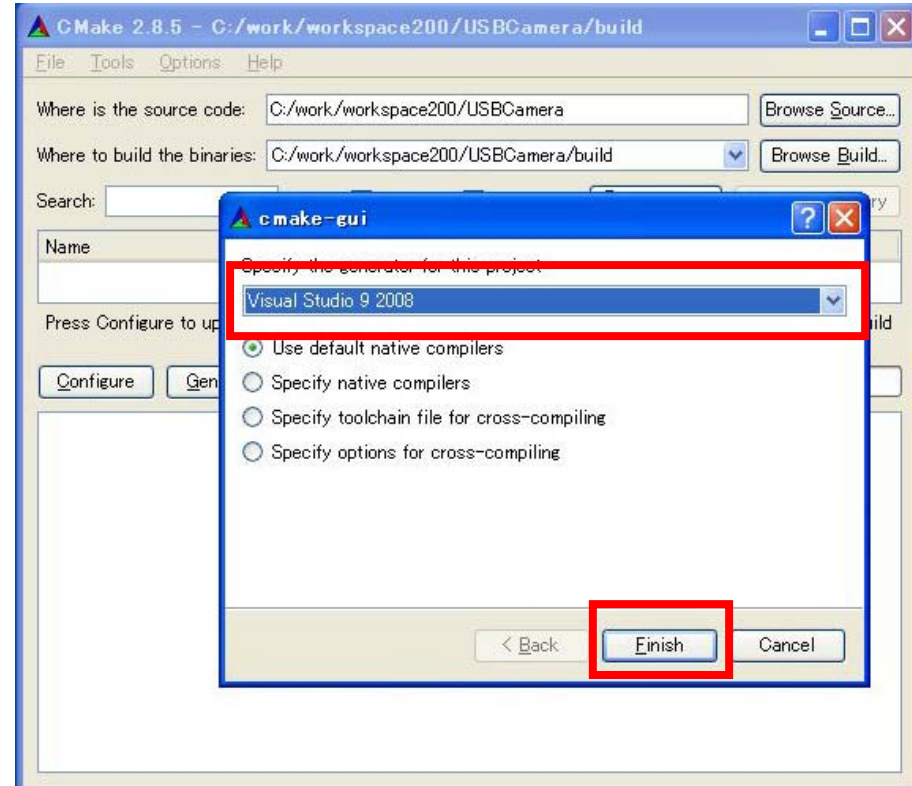
    return RTC::RTC_OK;
}
```

コンパイル(Windows,CMake利用)

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



② 「Configure」を実行し, 使用するプラットフォームを選択

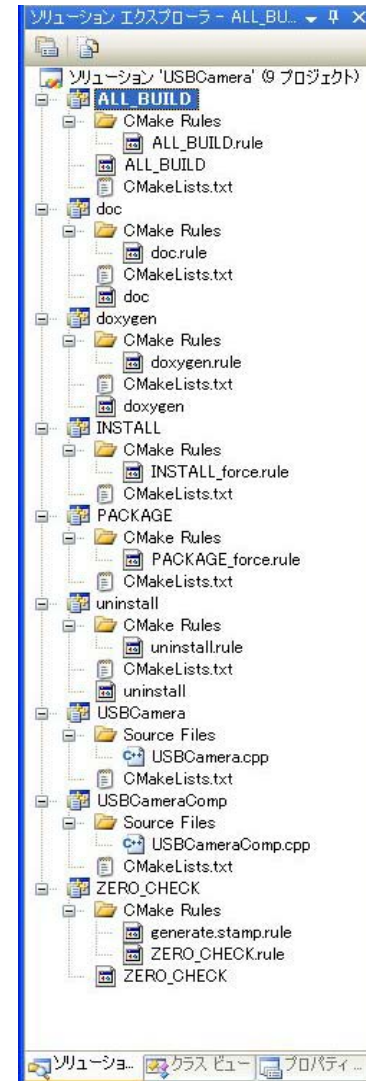
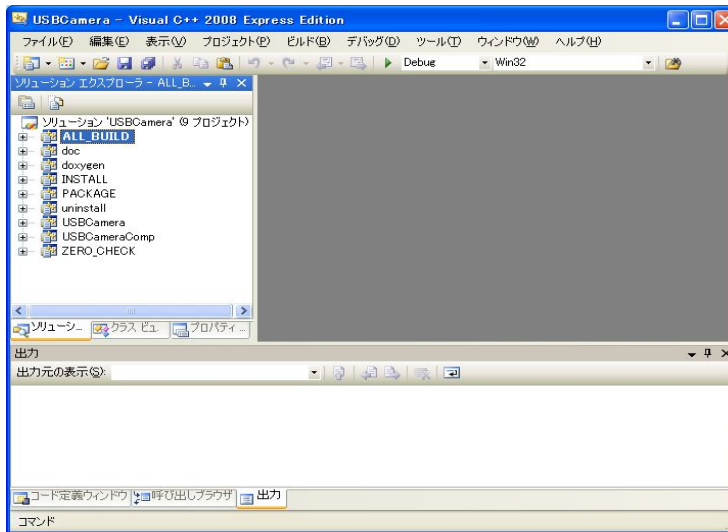
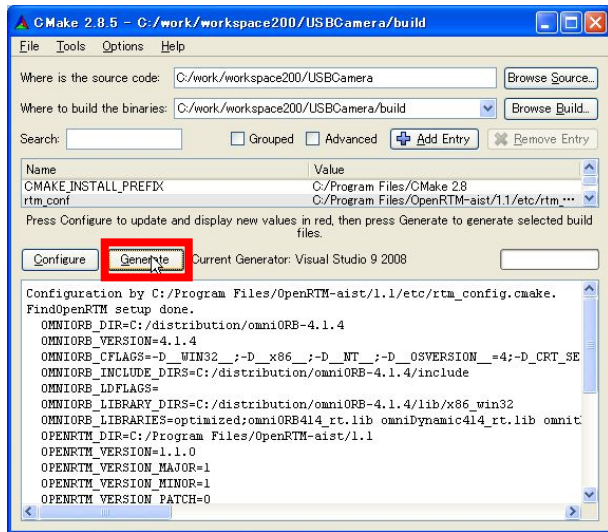


※binaryには, sourceとは別のディレクトリを指定する事を推奨

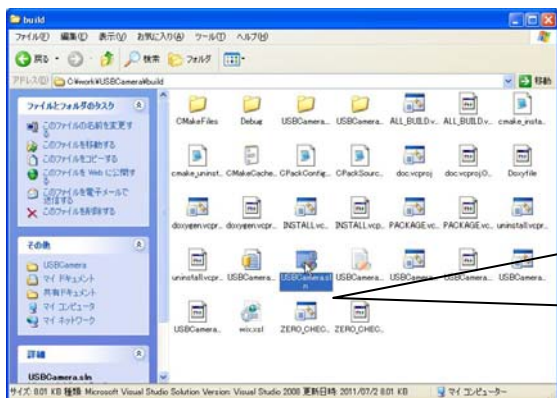
※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

コンパイル(Windows, CMake利用)

③ 正常終了後、「Generate」を実行

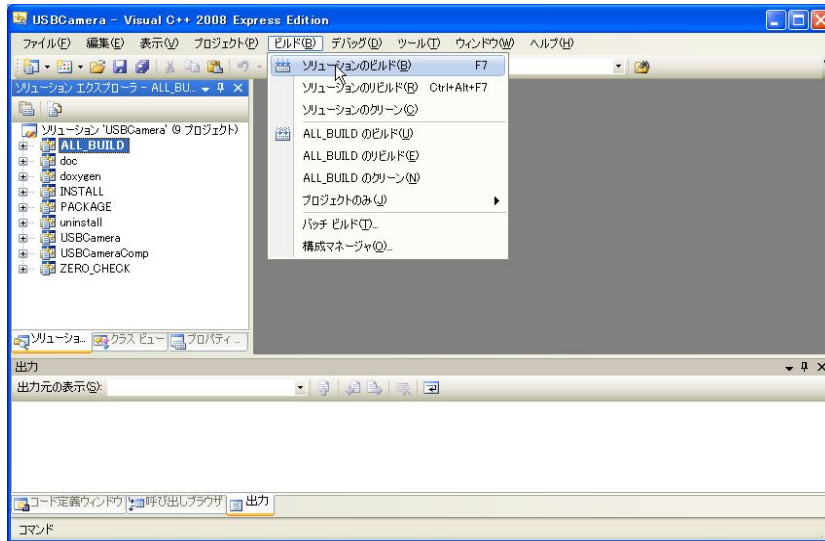


④ binaryとして指定したディレクトリ内にあるソリューションファイルを開く

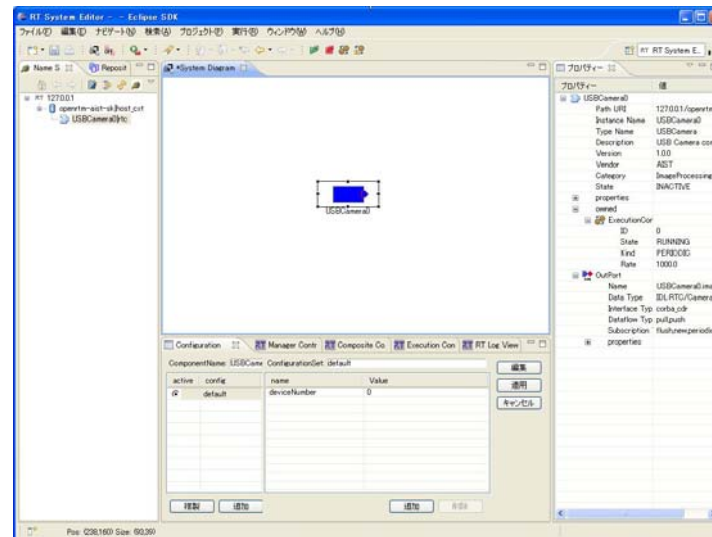
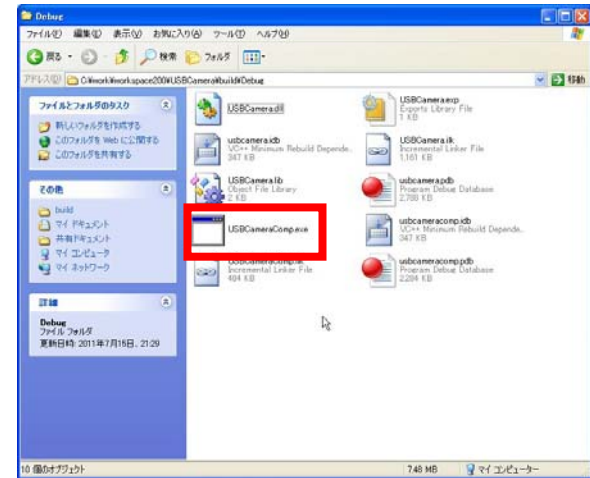


コンパイル・実行(Windows, CMake利用)

⑤ ソリューションをビルド



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



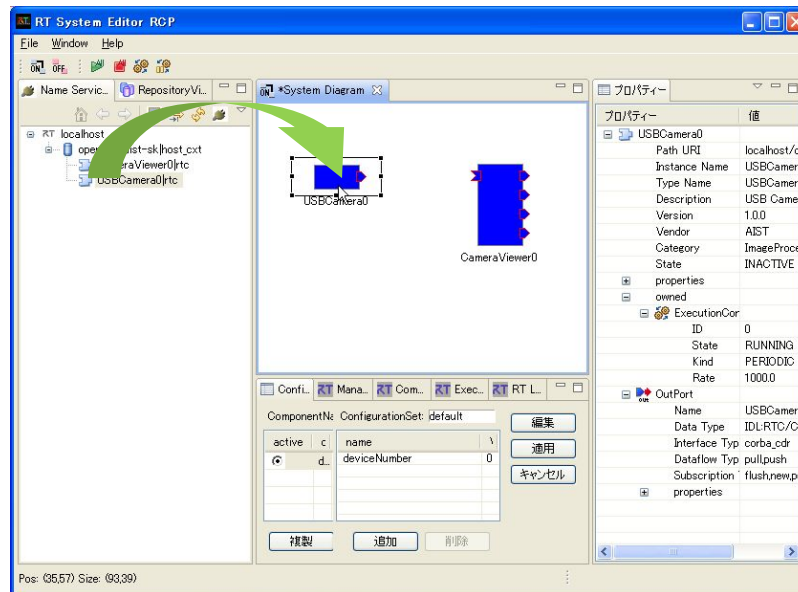
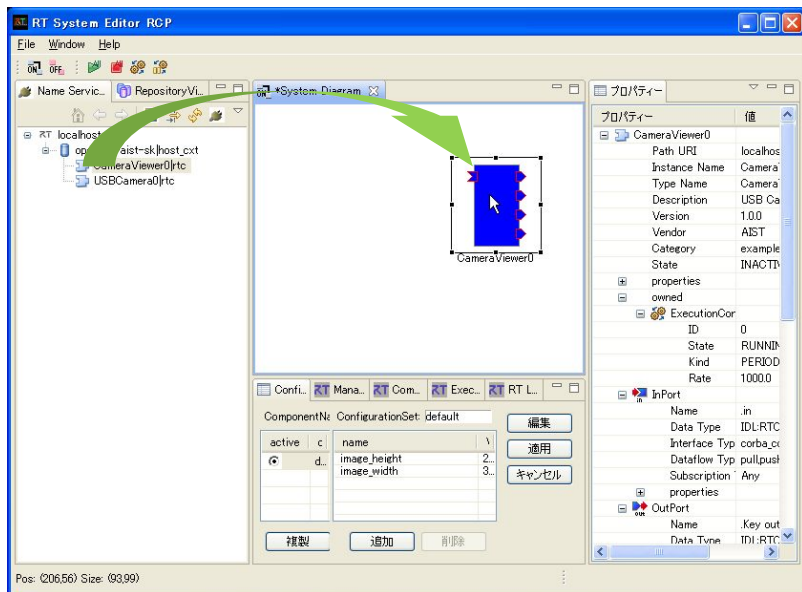
1. CameraViewerの起動

[スタート]メニューから起動

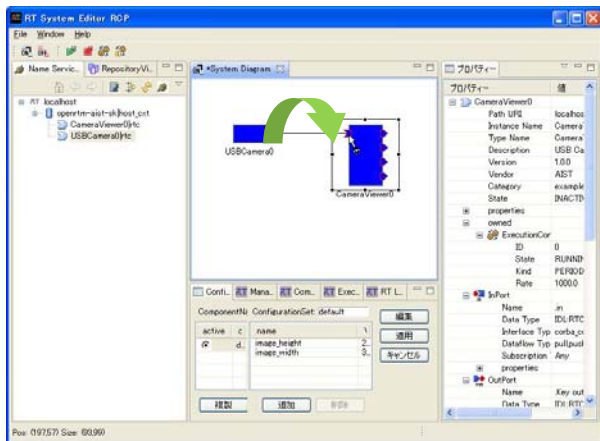
[プログラム]→[OpenRTM-aist 1.1]→[C++]→[components]
→[opencv-rtcs]→ [CameraViewerComp.exe]

2. コンポーネントの接続

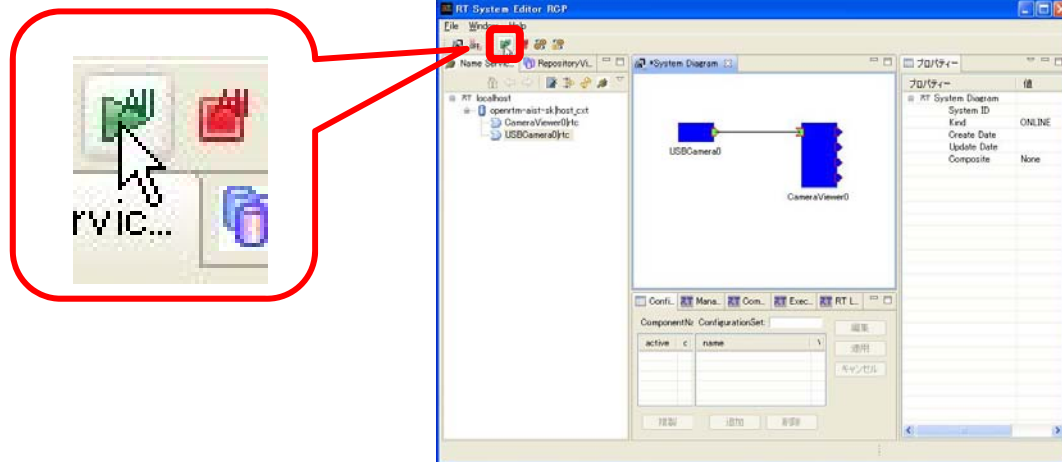
USBCameraとCameraViewerをシステムダイアグラムにドラッグ&ドロップ
します。



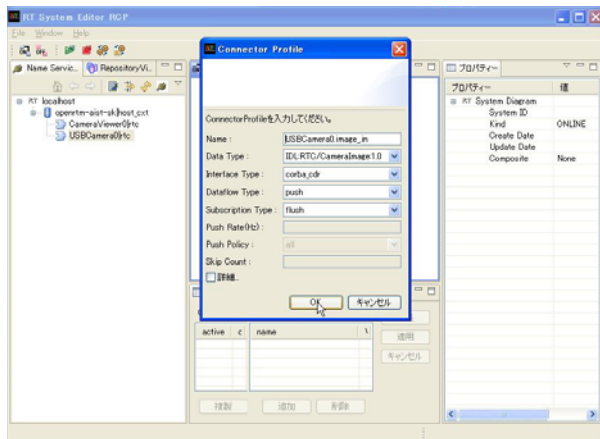
3. ポートを接続します。



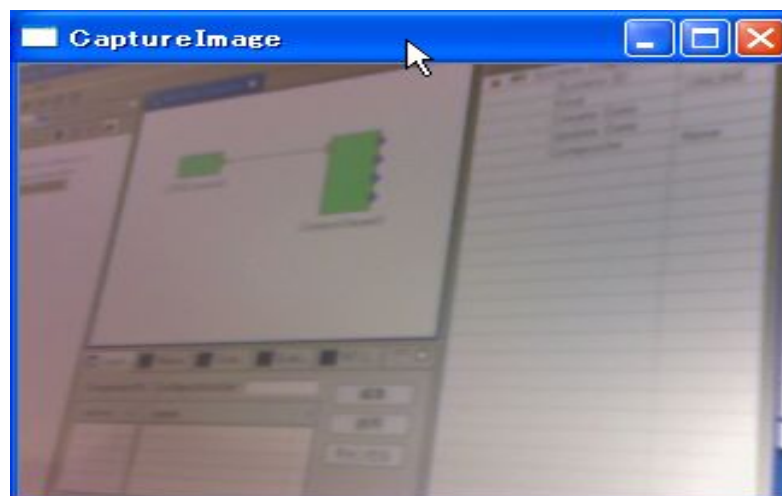
5. コンポーネントをアクティベートします。



4. 接続ダイアログでOKをクリックしポートを接続します。

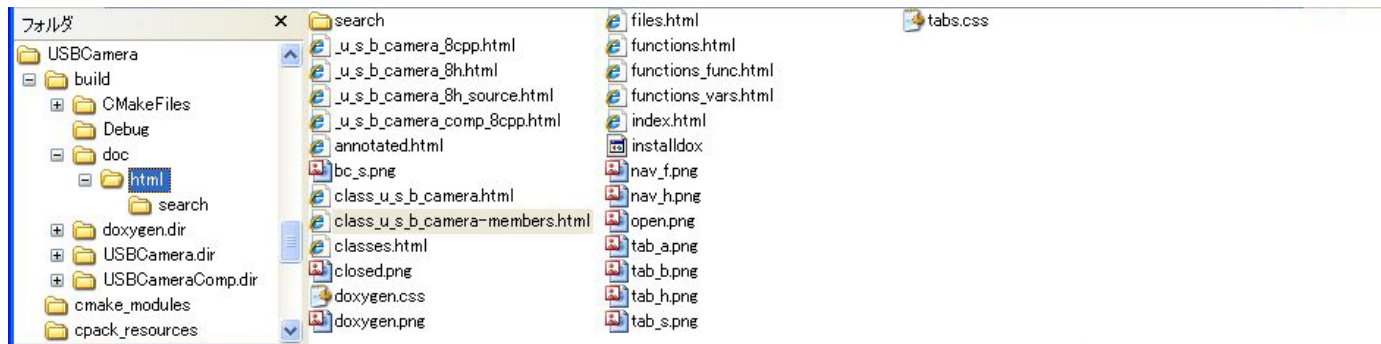


6. CameraViewerに画像が表示されます。



ドキュメント作成 (Windows, CMake利用)

※binaryにて指定したディレクトリ以下のdoc/html以下にdoxygenにて生成したドキュメント



生成されたドキュメントの例

usbcamera 1.0.0

メインページ クラス ファイル

検索

生成 生成索引 生成メソッド

クラス USBCamera

USB Camera component. [詳細]

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

すべてのメンバー一覧

Public メソッド

戻り値	メソッド名	説明
RTC::ReturnCode_t	onActivated (RTC::UniqueId ec_id)	
RTC::ReturnCode_t	onDeactivated (RTC::UniqueId ec_id)	
RTC::ReturnCode_t	onExecute (RTC::UniqueId ec_id)	

Protected 変数

型	変数名	説明
int	m_deviceNumber	
CameraImage	m_m_image	
OutPort< CameraImage >	m_m_imageOut	

説明

このクラスの説明は次のファイルから生成されました:

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

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

usbcamera 1.0.0

メインページ クラス ファイル

ファイル一覧

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

説明を見る。

```
00001 // -*- C++ -*-
00002 #ifndef USBCAMERA_H
00003 #define USBCAMERA_H
00004
00005 #include <rtm/Manager.h>
00006 #include <rtm/DataFlowComponentBase.h>
00007 #include <rtm/CorbaPort.h>
00008 #include <rtm/DataInPort.h>
00009 #include <rtm/DataOutPort.h>
00010 #include <rtm/idl/BasicDataTypesSkel.h>
00011 #include <rtm/idl/ExtendedDataTypesSkel.h>
00012 #include <rtm/idl/InterfaceDataTypesSkel.h>
00013
00014 // Service implementation headers
00015 #ifndef <rtc-template block="service_impl_h">
00016
00017 #endif
00018
00019 // Service Consumer stub headers
00020 #ifndef <rtc-template block="consumer_stub_h">
00021
00022 #endif
00023
00024 #endif
00025
00026 using namespace RTC;
00027
00028 class USBCamera
00029 : public RTC::DataFlowComponentBase
00030 {
00031 public:
00032     USBCamera (RTC::Manager* manager);
00033     ~USBCamera ();
00034
00035     // <rtc-template block="public_attribute">
```

動作確認(コンポーネントの追加)

1. Flipコンポーネントの起動

[スタート]メニューから起動

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

2. コンポーネントの追加

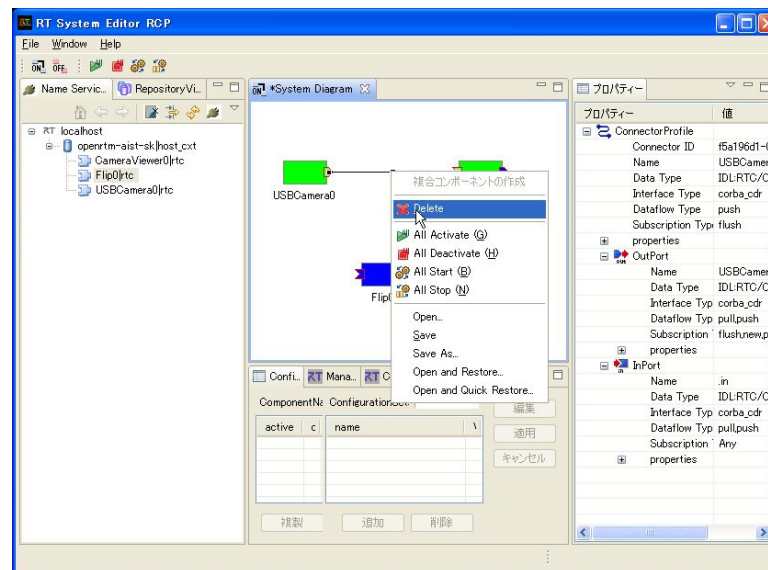
Flipをシステムダイアグラムにドラッグ & ドロップします。

3. USBCameraとCameraViewerのポートを切断します。

(1) ポートの接続線をクリックします。

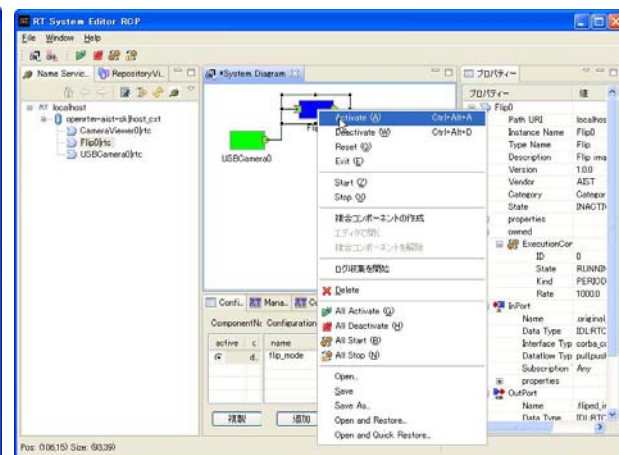
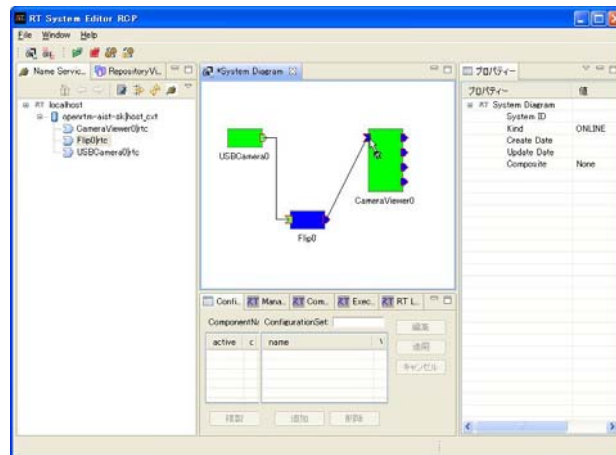
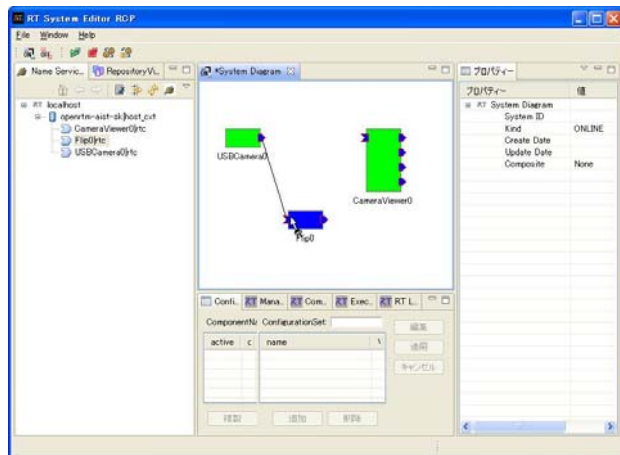
(2) 接続線を右クリックします。

(3) “Delete”をクリックします。



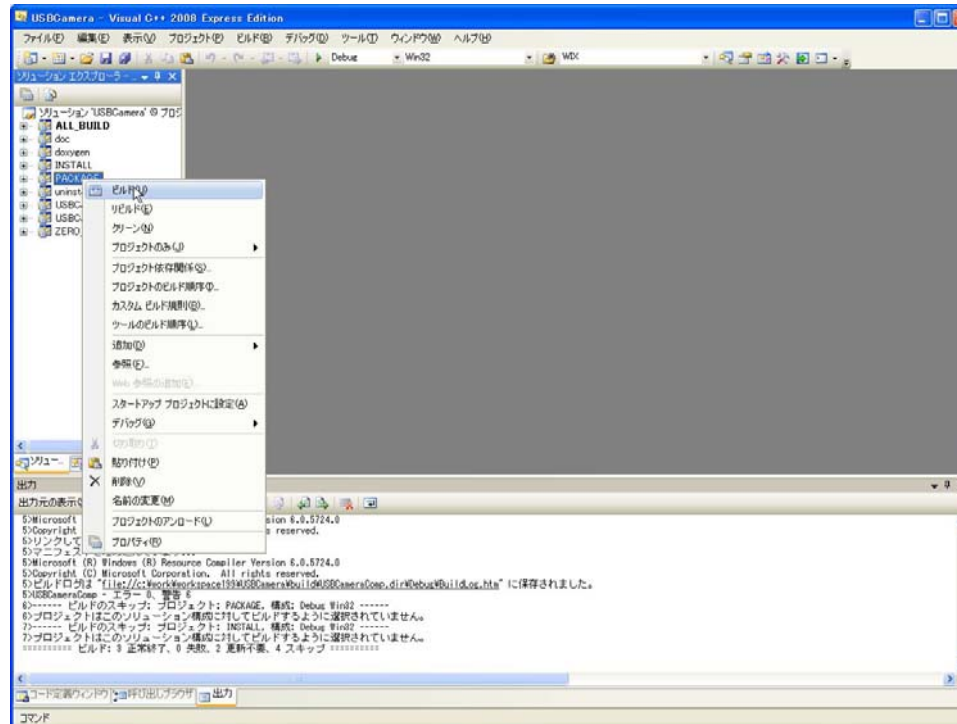
動作確認(コンポーネントの追加)

1. コンポーネントの接続
USBCameraのOutPortとFlipのInPortを接続します。
2. コンポーネントの接続
FlipのOutPortとCameraViewerのInPortを接続します。
3. Flipコンポーネントをアクティベートします。
 - (1) Flipを右クリックします。
 - (2) 表示されたメニューにて“Activate(A)”をクリックします。



配布用パッケージ作成(Windows,CMake利用)

■ ソリューション中の「PACKAGE」をビルド



● binaryにて指定したディレクトリ直下にmsi形式のインストールパッケージを生成

● コンポーネントのインストール先

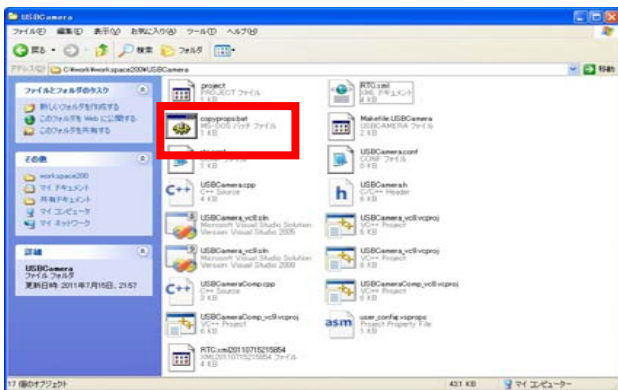
C:\Program Files\OpenRTM-aist\1.1\components\<言語>\<パッケージ名>

RTCBuilderの補足

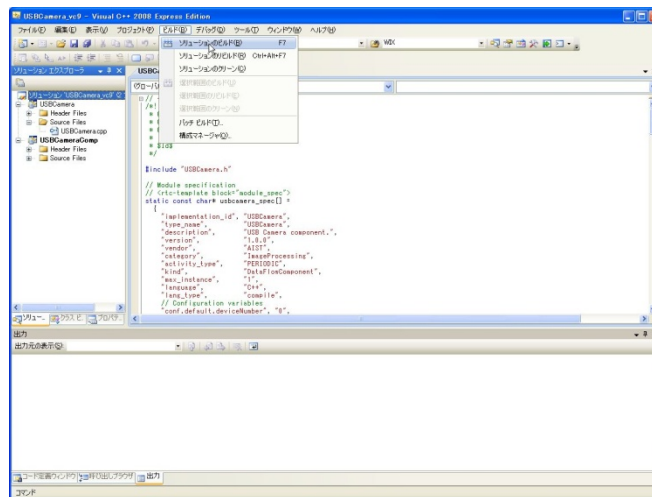


コンパイル・実行(Windows)

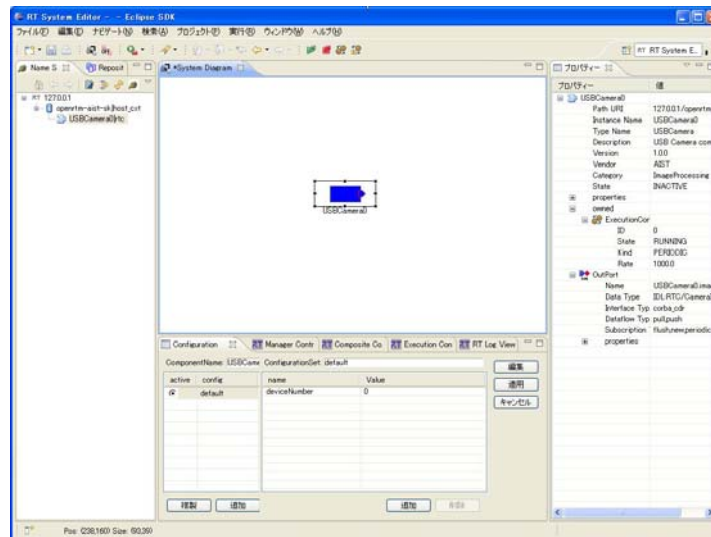
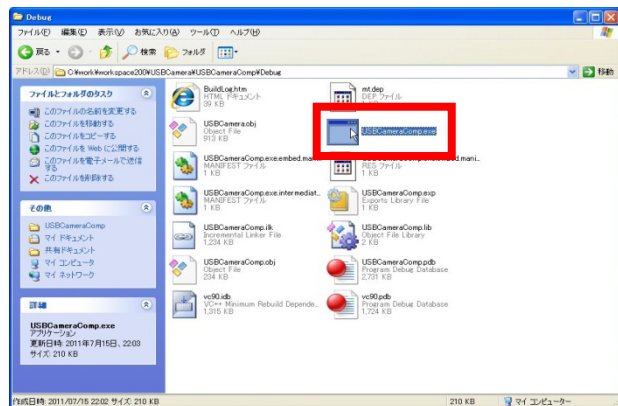
①コード生成先ディレクトリ内の「copyprops.bat」をダブルクリックして、設定ファイルをコピー



②VisualStudioを用いたビルド

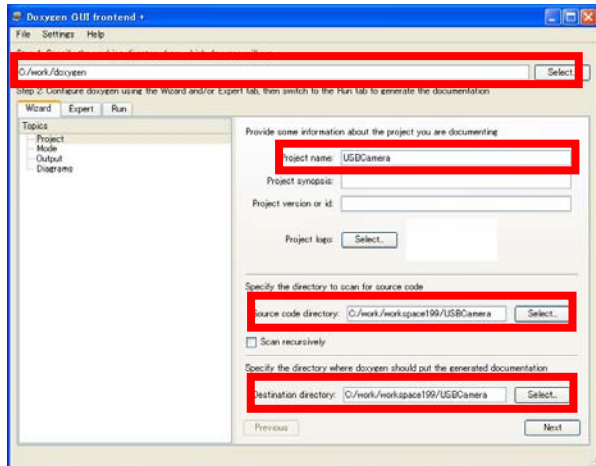


③USBCameraComp¥¥Debug内のUSBCameraComp.exeを起動

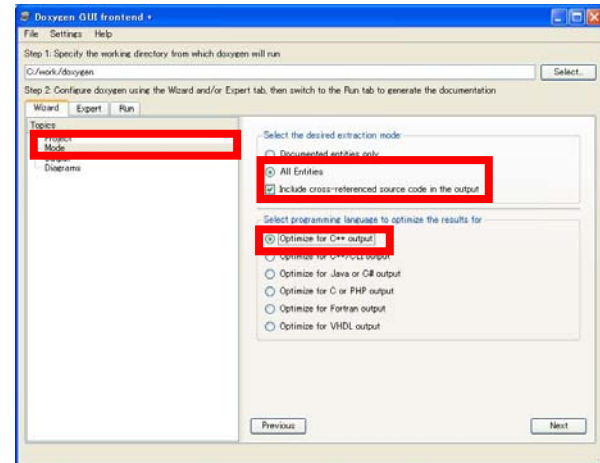


ドキュメント作成(Windows)

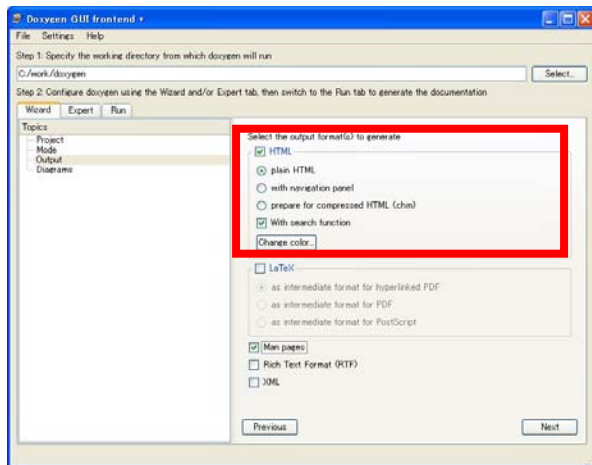
- ① Doxygen用GUIツールを起動
作業用ディレクトリ,ソース格納場所,
生成ファイル出力先,プロジェクト名を指定



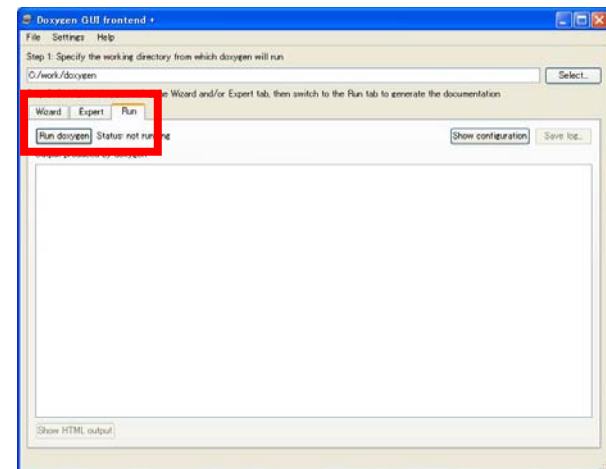
- ② 「Mode」セクションにて,
出力内容,使用言語を指定



- ③ 「Output」セクションにて, html出力を指定



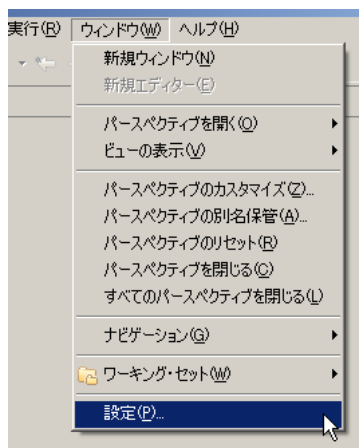
- ③ 「Run」タブにて, 「Run doxygen」を実行



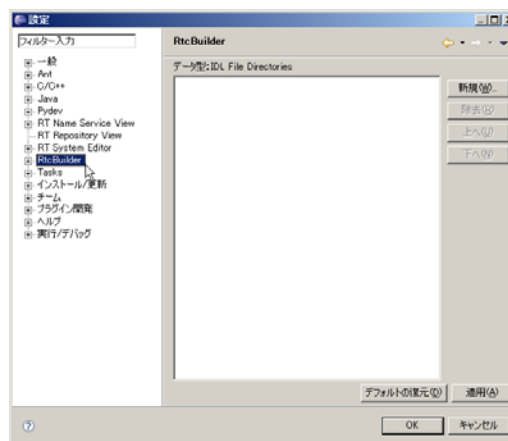
■ DataPortにて利用するデータ型の指定

→データ型を定義したIDLファイルが格納されているディレクトリを指定

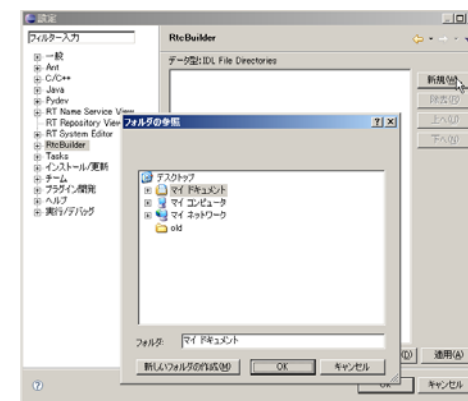
①メニューから
「ウィンドウ」→「設定」



②「RtcBuilder」を選択



③「新規」ボタンにて表示される
ディレクトリ選択ダイアログ
にて場所を指定



※独自に定義したデータ型を使用する場合のみ必要な設定

OpenRTM-aistにて標準で用意されている型のみを使用する場合には設定不要

・標準型の定義内容格納位置：[RTM_Root]rtm/idl

→BasicDataType.idl, ExtendedDataTypes.idlなど

→デフォルト設定では，[RTM_Root]=C:/Program Files/OpenRTM-aist/1.1/

青梅商工会議所主催 RTM講習会

日時:2011年7月25日(月) 10:30~17:30

場所:産業技術総合研究所 中央第2 本部情報棟1F ネットワーク会議室

