PFCore(RTミドルウェア)トレーニング 中級編

日時：2013年1月22日(火)　10:00～17:00
場所：大阪大学　吹田キャンパス　産学連携本部B棟1F会議室

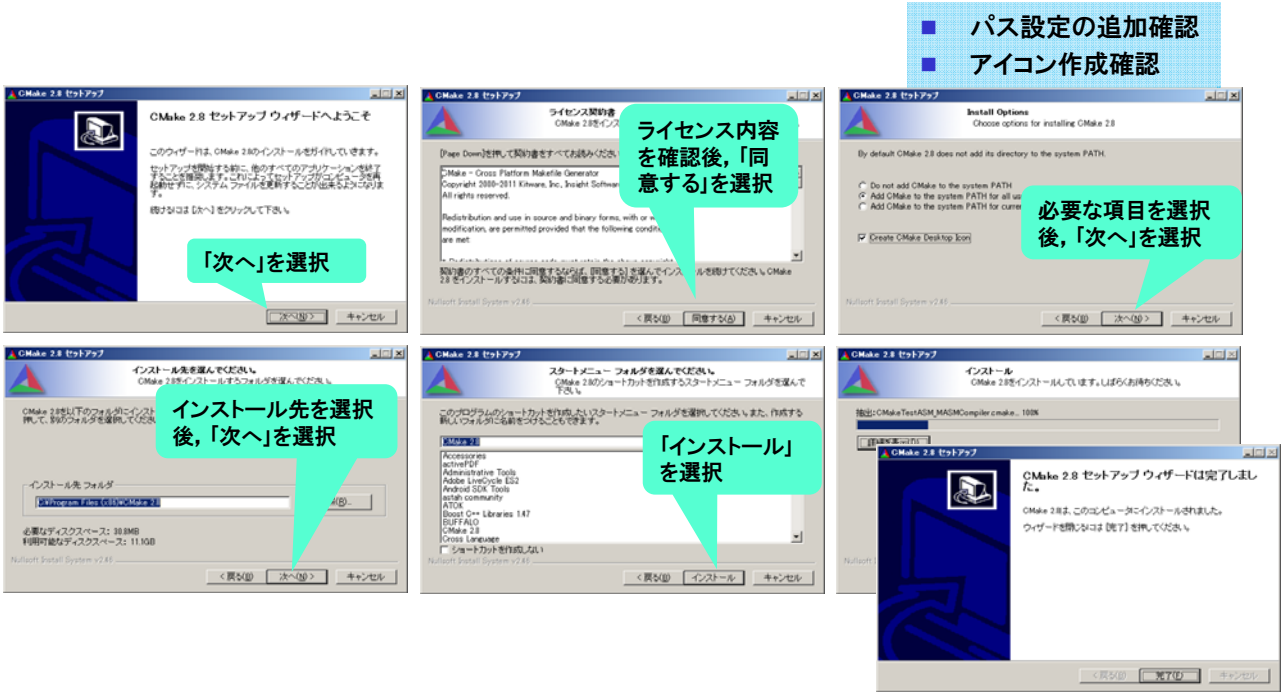
PFCore(RTミドルウェア)トレーニング 中級編

10:00-11:00	**第1部：RTコンポーネントプログラミングの概要**
	担当：安藤慶昭（産業技術総合研究所）
	概要:RTコンポーネントの作成方法，設計時の注意点などの概要について解説します。
11:00-12:00	**第2部：RTミドルウェア(PFcore)開発支援ツールとRTコンポーネントの作成方法**
	担当：坂本 武志（株式会社 グローバルアシスト）
	概要:RTコンポーネントを開発するために必要なツールのインストール方法，標準ツールRTCBulderを使用して、RTコンポーネントを開発する方法の概略を説明します.
12:00-13:00	**休憩**
13:00-17:00	**第3部：RTコンポーネント開発実習**
	担当：安藤慶昭（産業技術総合研究所）
	概要:OpenRTM-aistでのコンポーネントの作成方法を実際に体験して頂きます．画像処理システムを対象にRTCBuilderを使用したRTコンポーネントの設計，実装を行います.

第2部　RTミドルウェア(PFcore)開発支援ツールと RTコンポーネントの作成方法

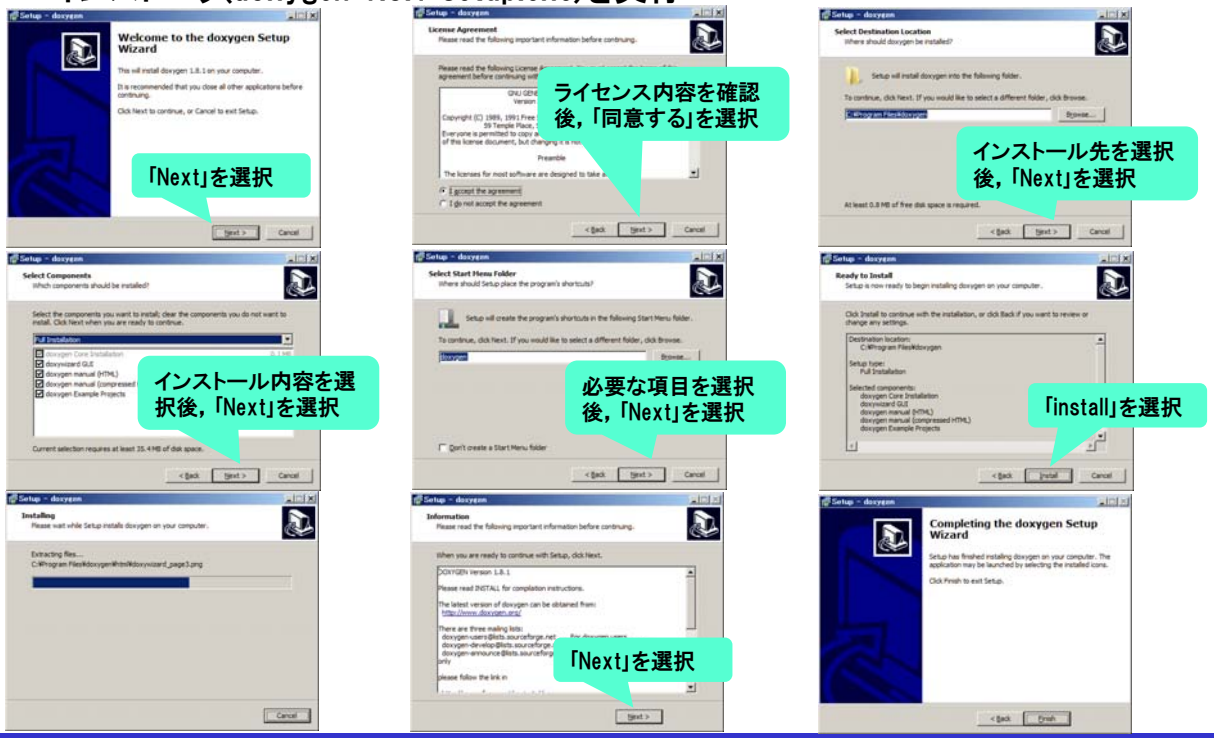
関連ツールのインストール

- CMake：Visual C++向けのプロジェクトファイルを作成
  - インストーラ(cmake-2.8.8-win32-x86.exe)を実行



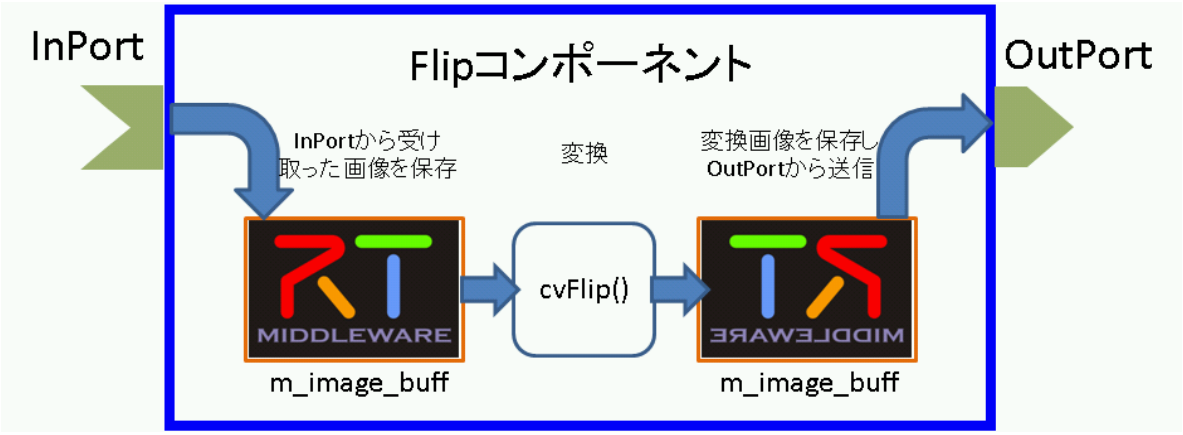
関連ツールのインストール

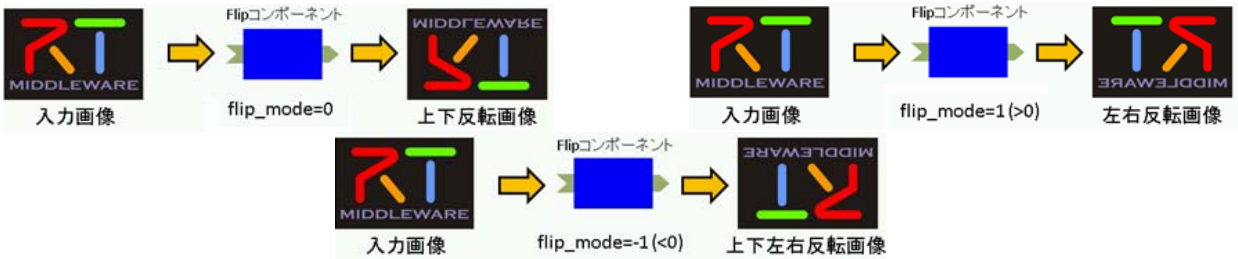
- Doxygen:ビルド中にドキュメント生成を実行
  - インストーラ(doxygen-1.8.1-setup.exe)を実行



作成するコンポーネントの概要

- Flipコンポーネント
  - OpenCVのcvFlip関数を利用して入力画像を反転して出力する





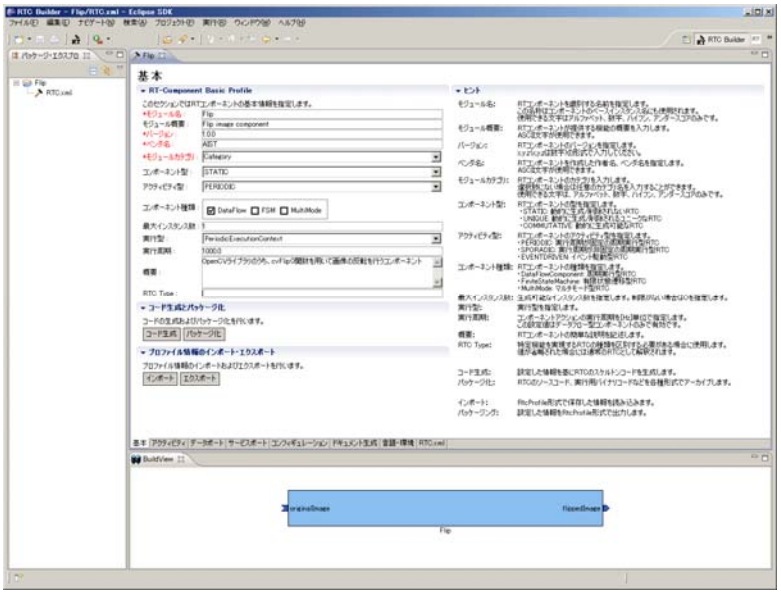
サンプルプロファイルのダウンロード

- 以下から「RTC.xml」をダウンロードします。
  http://openrtm.org/openrtm/ja/tutorial/jst_osaka_20130121

RTCBuilderについて

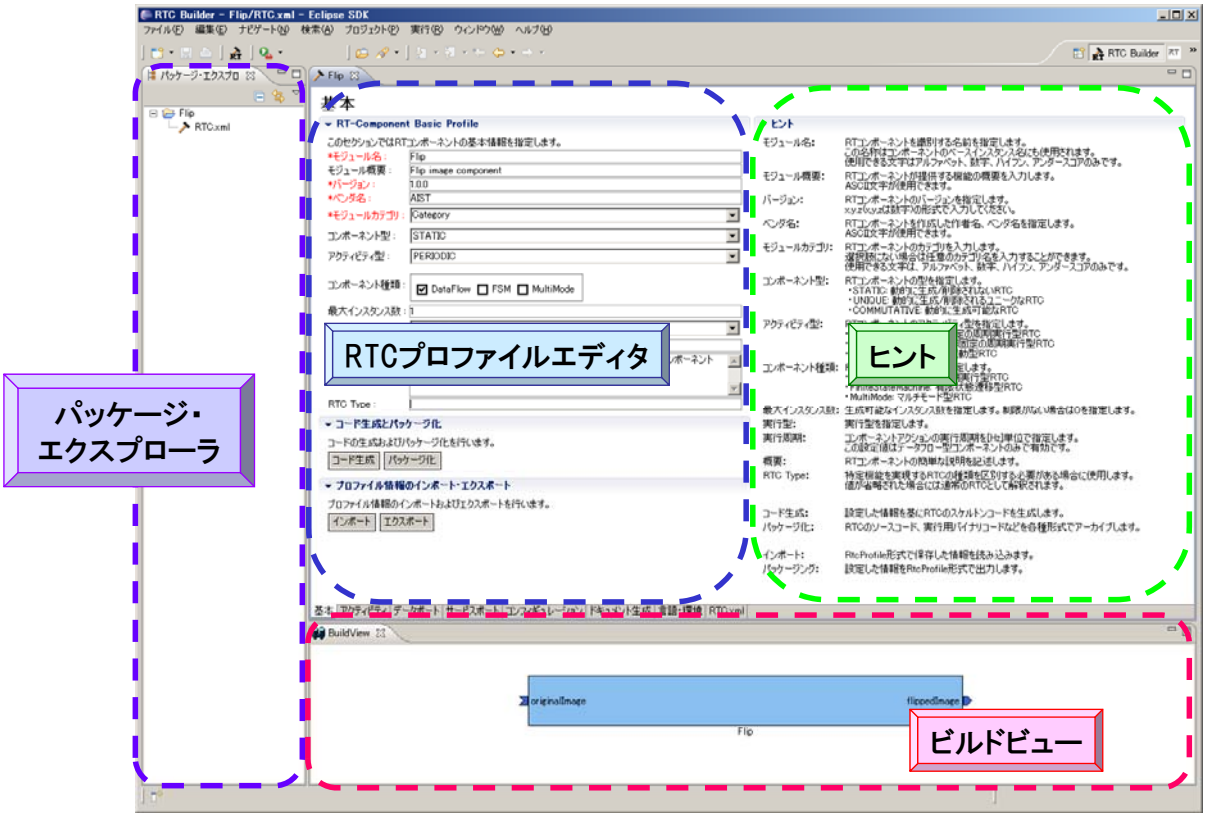
RTCBuilder概要

- 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

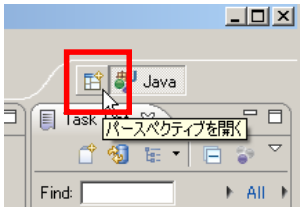
画面構成

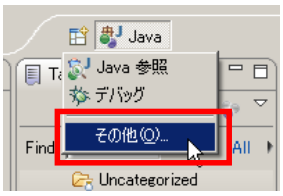


準備

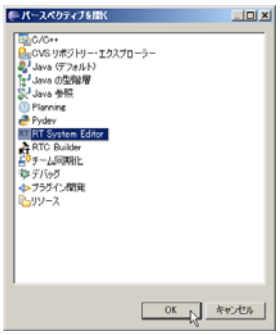
■ パースペクティブの切り替え

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





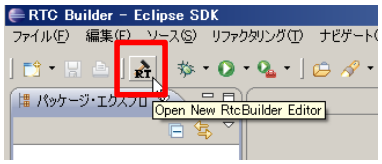
②一覧画面から対象ツールを選択



※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

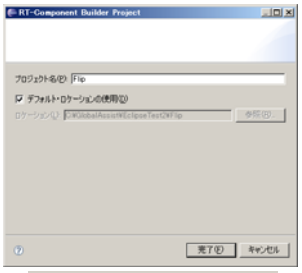


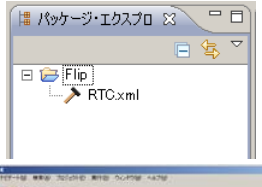
※メニューから「ファイル」-「新規」-「プロジェクト」を選択
【新規プロジェクト】画面にて「その他」-「RtcBuilder」を選択
し，「次へ」
※メニューから「ファイル」-「Open New Builder Editor」を選択

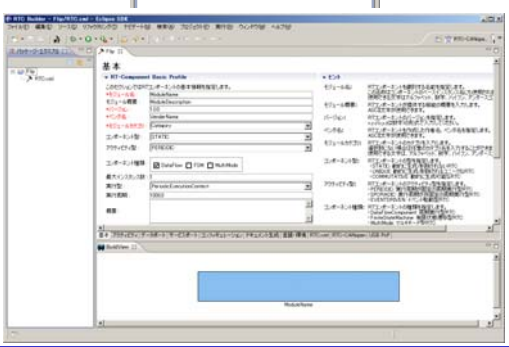
※任意の場所にプロジェクトを作成したい場合
②にて「デフォルト・ロケーションの使用」チェックボックス
を外す
「参照」ボタンにて対象ディレクトリを選択
→物理的にはワークスペース以外の場所に作成される
論理的にはワークスペース配下に紐付けされる

プロジェクト名：Flip

②「プロジェクト名」欄に入力し，「終了」

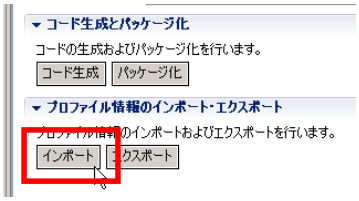




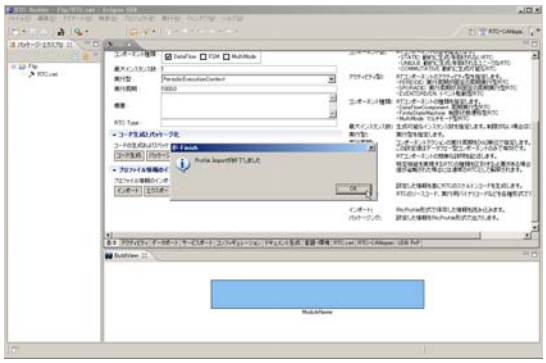


プロファイル インポート

①「基本」タブ下部の「インポート」ボタンをクリック



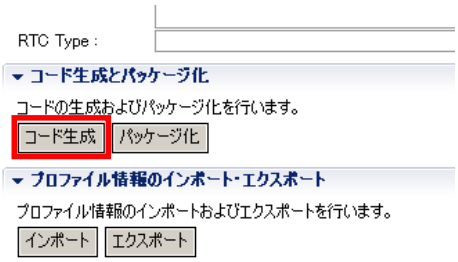
②【インポート】画面にて対象ファイルを選択

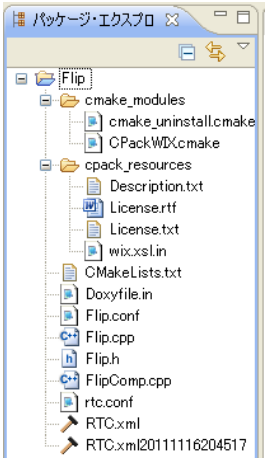


- 作成済みのRTコンポーネント情報を再利用
  - 「エクスポート」機能を利用して出力したファイルの読み込みが可能
  - コード生成時に作成されるRtcProfileの情報を読み込み可能
  - XML形式，YAML形式での入出力が可能

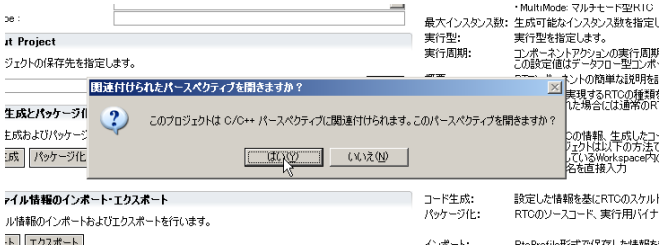
コード生成

- コード生成





- 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）

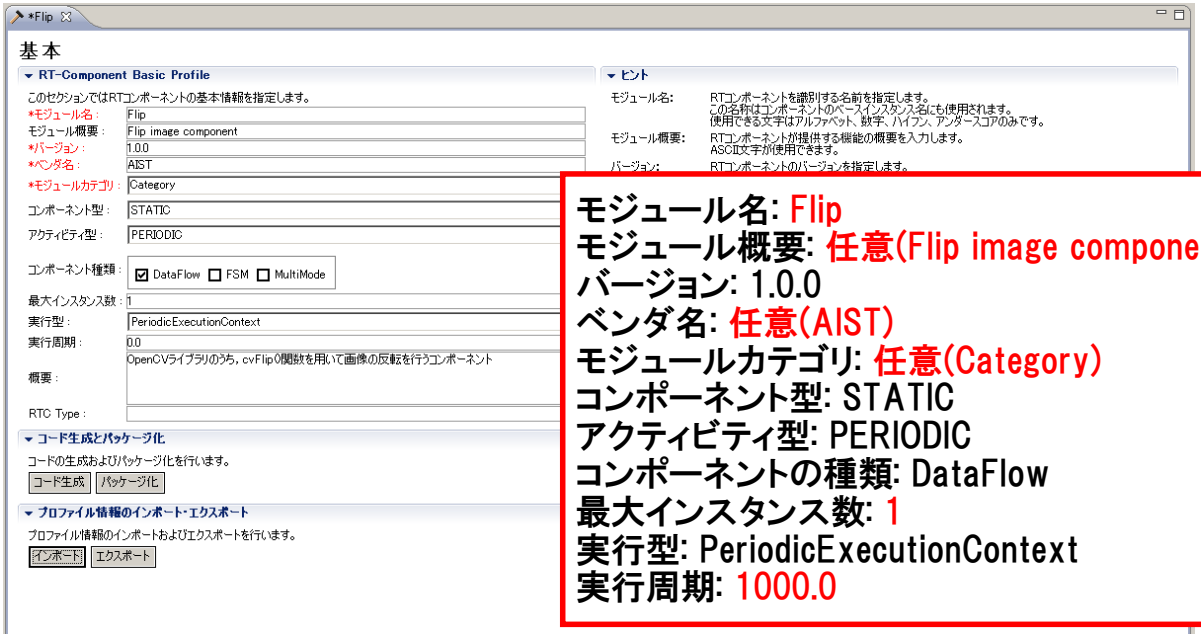
Python版 → PyDev

RTCプロファイルエディタ

画面要素名	説明
基本プロファイル	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

基本プロファイル

- 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: Flip
モジュール概要: 任意(Flip image 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(Category)
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

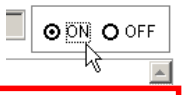
※エディタ内の項目名が赤字の要素は必須入力項目
※画面右側は各入力項目に関する説明

アクティビティ・プロファイル

■ 生成対象RTCで実装予定のアクティビティを設定

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



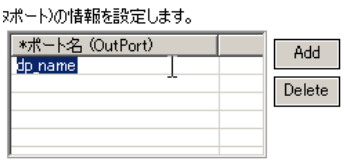
以下をチェック：
onActivated
onDeactivated
onExecute

※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

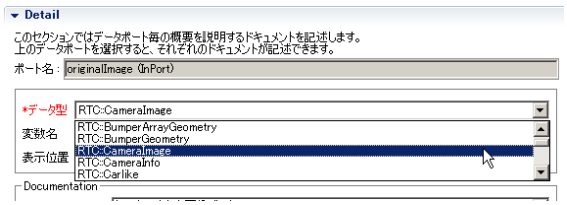
データポート・プロファイル

■ 生成対象RTCに付加するDataPortの情報を設定

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



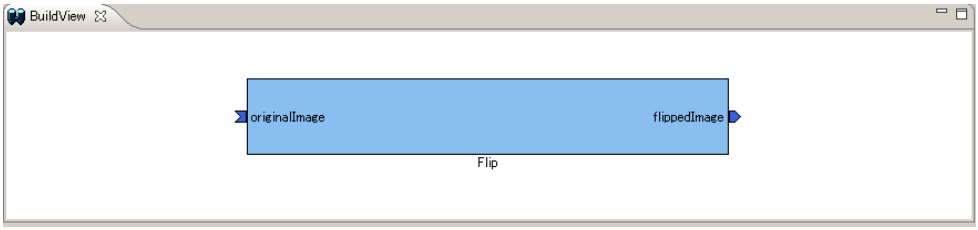
②設定する型情報を一覧から選択



※データ型は，型定義が記載されたIDLファイルを設定画面にて追加することで追加可能
※OpenRTM-aistにて事前定義されている型については，デフォルトで使用可能
→[RTM_Root]rtm/idl 以下に存在するIDLファイルで定義された型
※各ポートに対する説明記述を設定可能
→記述した各種コメントは，生成コード内にDoxygen形式で追加される

データポート・プロファイル

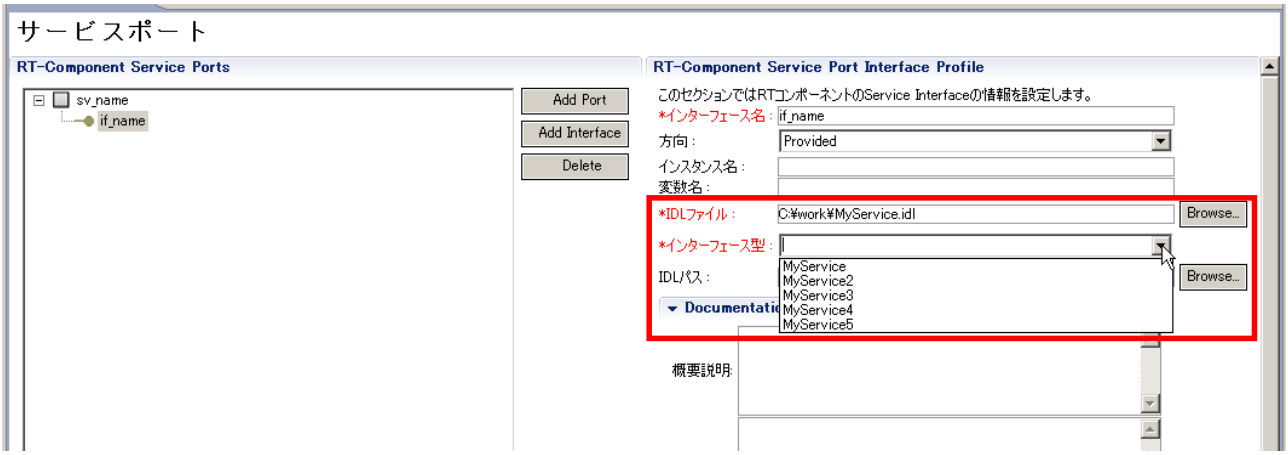
※Portの設定内容に応じて，下部のBuildViewの表示が変化



● InPort
ポート名: originalImage
データ型: RTC::CameraImage
変数名: originalImage
表示位置: left
● OutPort
ポート名: flippedImage
データ型: RTC::CameraImage
変数名: flippedImage
表示位置: right

サービスポート・プロファイル

■ 生成対象RTCに付加するServicePortの情報を設定

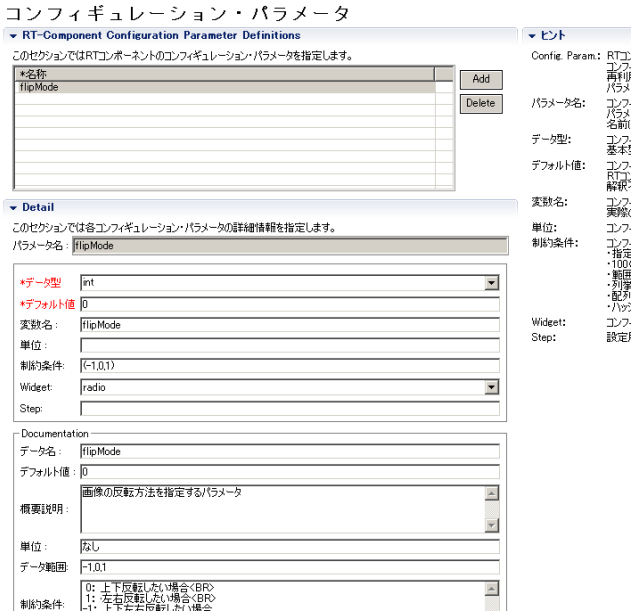


■ サービスインターフェースの指定
● IDLファイルを指定すると，定義されたインターフェース情報を表示

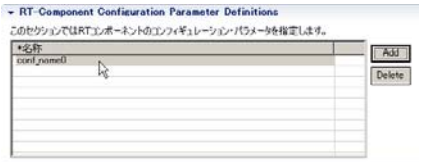
今回のサンプルでは未使用

コンフィギュレーション・プロファイル

■ 生成対象RTCで使用する設定情報を設定



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: flipMode
データ型: int
デフォルト値: 0
変数名: flipMode
制約条件: (-1, 0, 1)
Widget: radio

※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，Widgetの設定方法

■ 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<=x<=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}

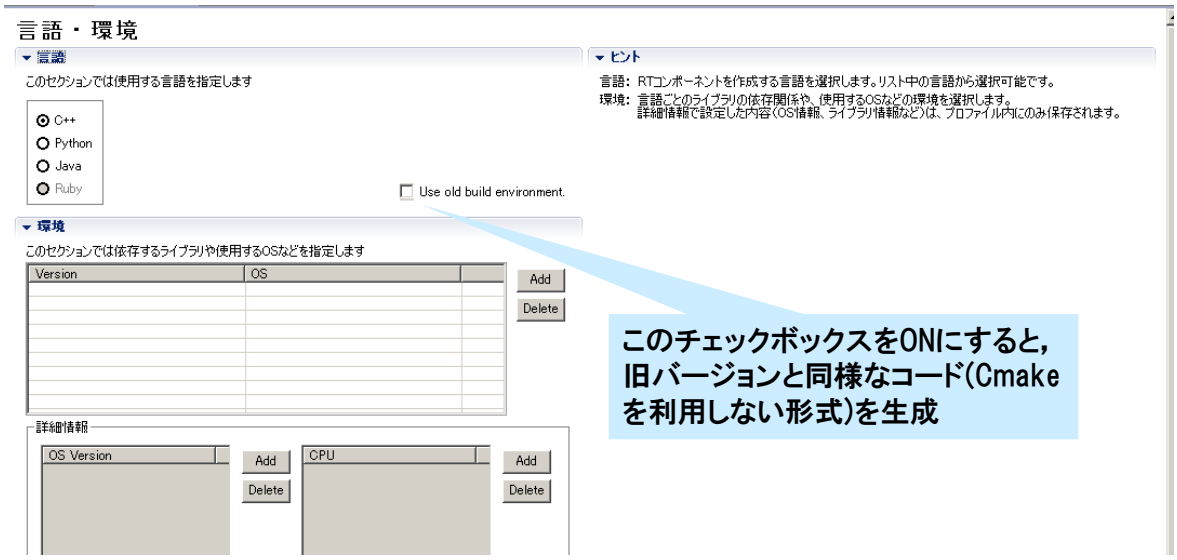
■ Widget

- 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

言語・環境・プロファイル

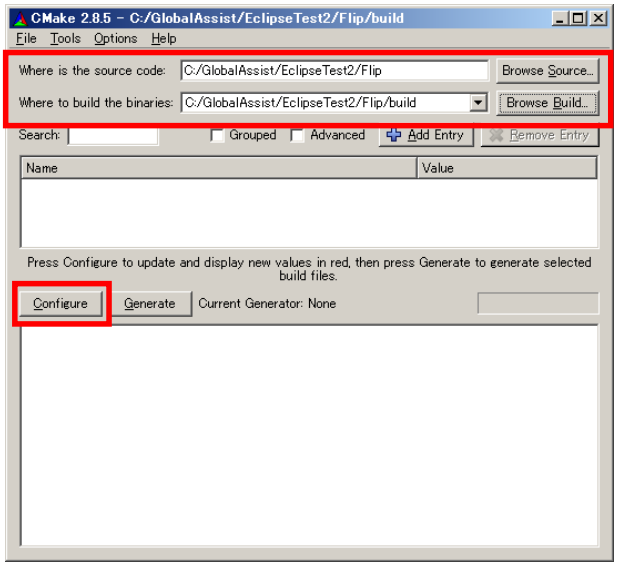
- 生成対象RTCを実装する言語，動作環境に関する情報を設定



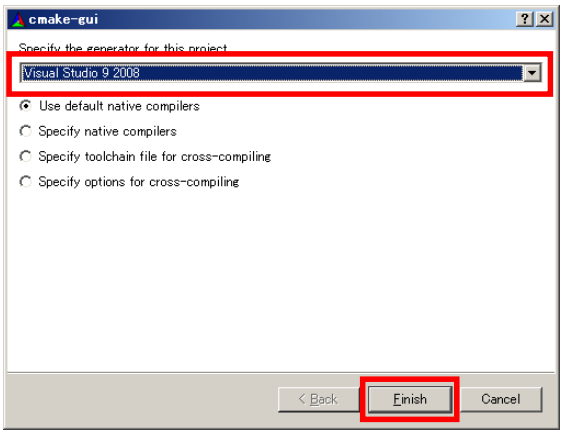
「C++」を選択

コンパイル(Windows,CMake利用)

①GUI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

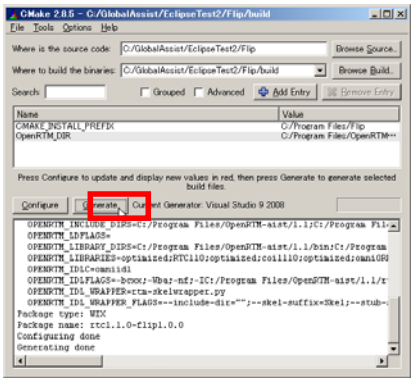


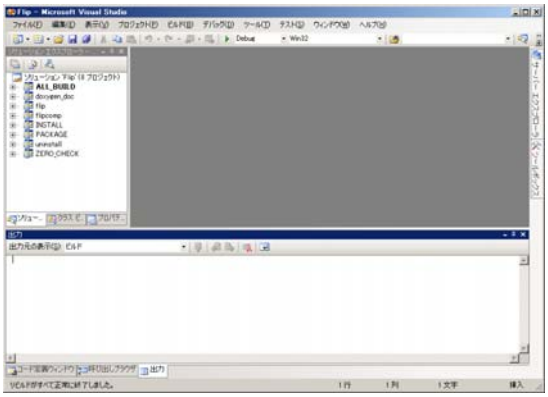
※binaryには，sourceとは別のディレクトリを指定する事を推奨

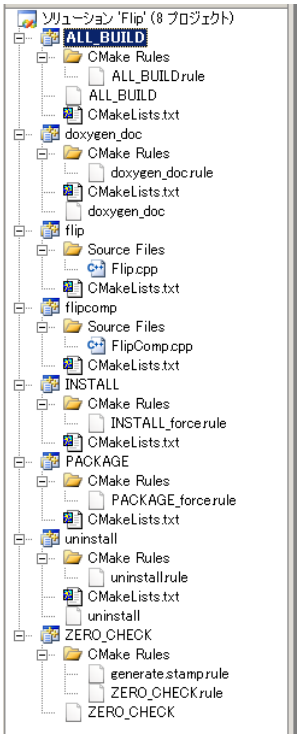
※日本語は文字化けしてしまうため英数字のみのディレクトリを推奨

コンパイル(Windows,CMake利用)

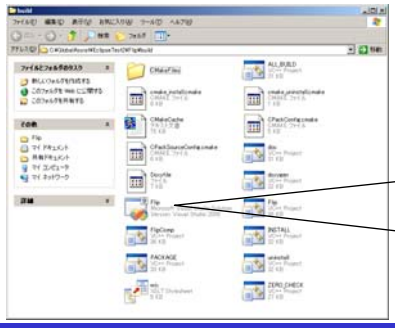
③正常終了後，「Generate」を実行





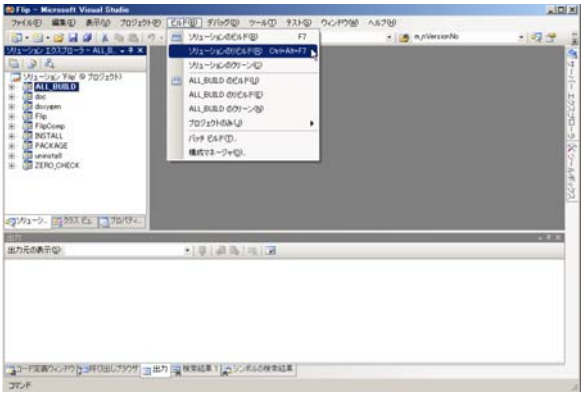


④binaryとして指定したディレクトリ内にある
ソリューションファイルを開く

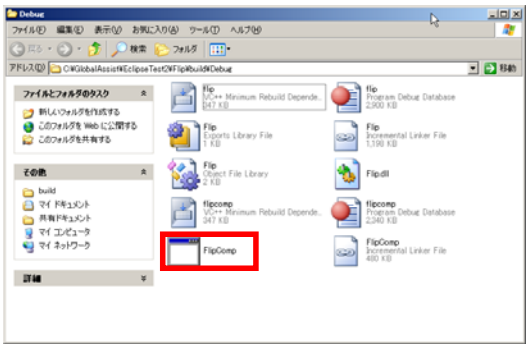


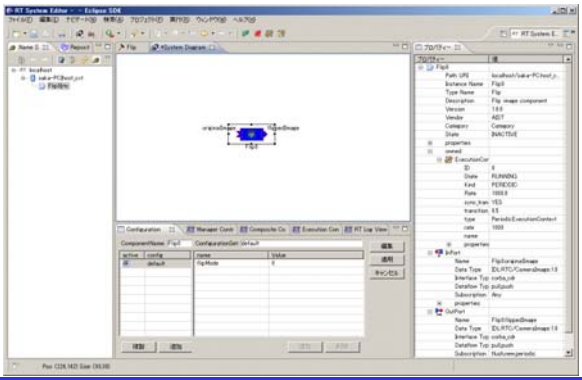
コンパイル・実行(Windows,CMake利用)

⑤ソリューションをビルド



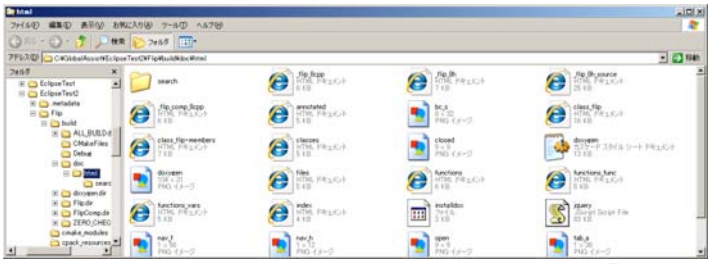
⑥binaryにて指定したディレクトリ以下の
Src/Debug内のFlipComp.exeを起動



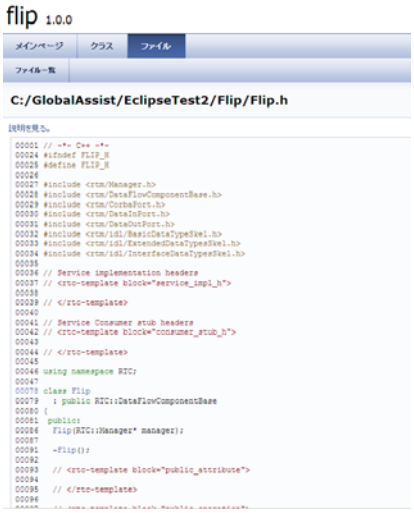


ドキュメント作成(Windows,CMake利用)

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

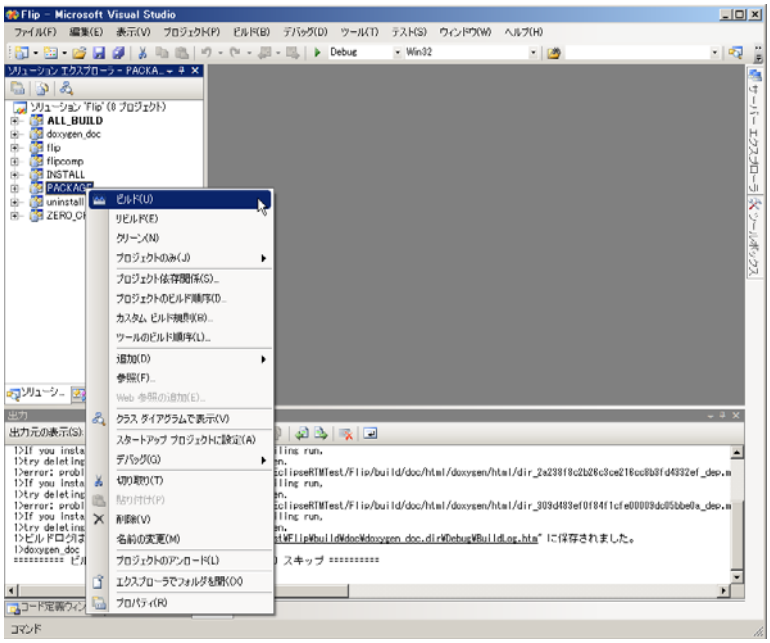


- 生成されたドキュメントの例



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

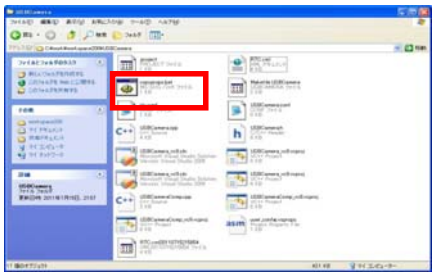


- binaryにて指定したディレクトリ直下にmsi形式のインストールパッケージを生成
  - コンポーネントのインストール先
    C:¥Program Files¥OpenRTM-aist¥1.1¥components¥<言語>/<パッケージ名>

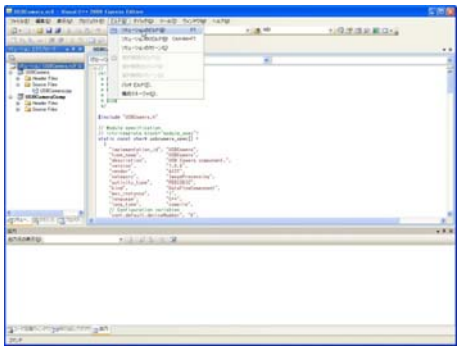
RTCBuilder補足説明

コンパイル・実行(Windows)

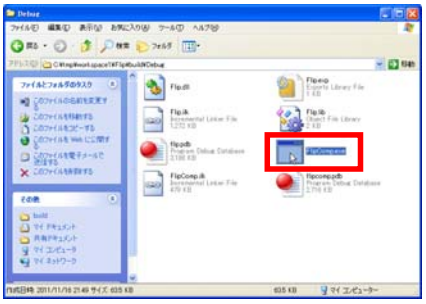
①コード生成先ディレクトリ内の
「copyprops.bat」をダブルクリックして，
設定ファイルをコピー

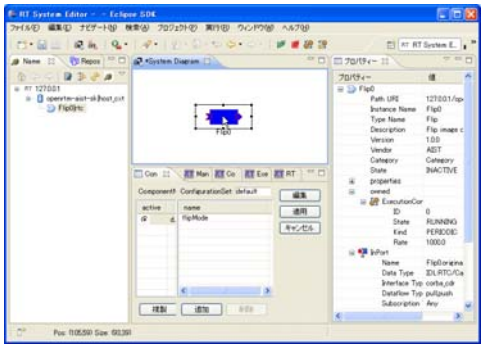


②VisualStudioを用いたビルド



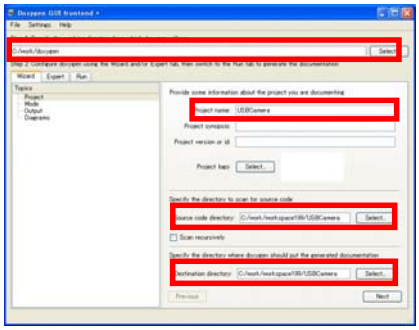
③FlipComp¥¥Debug内の
FlipComp.exeを起動



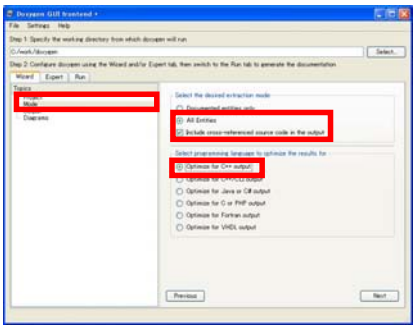


ドキュメント作成(Windows)

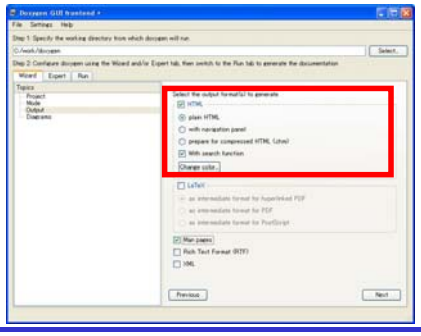
①Doxygen用GUIツールを起動
作業用ディレクトリ,ソース格納場所,
生成ファイル出力先,プロジェクト名を指定



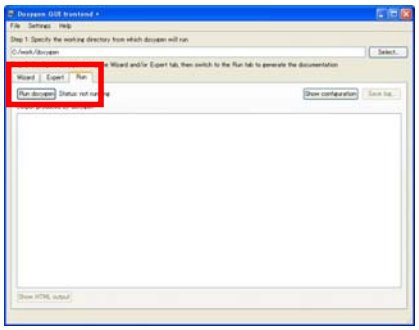
②「Mode」セクションにて,
出力内容,使用言語を指定



③「Output」セクションにて，html出力を指定



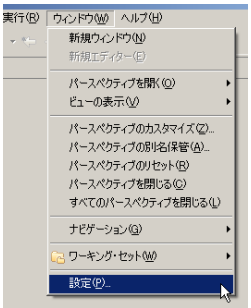
③「Run」タブにて，「Run doxygen」を実行



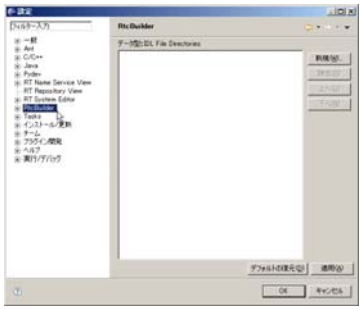
各種設定

- DataPortにて利用するデータ型の指定
  →データ型を定義したIDLファイルが格納されているディレクトリを指定

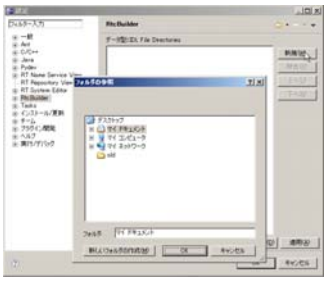
①メニューから
「ウインドウ」ー「設定」



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/