第3部：プログラミング演習

名城大学
理工学部メカトロニクス工学科
大原賢一

第3部での目標

- Flipコンポーネントを自身で作成することで，RTコンポーネント開発の基礎を習得すること.
- ここでは，第2部で用いたFlipコンポーネントのひな形を用います.

CMakeLists.txtの修正

- 出力された以下のフォルダの中にあるCMakeLists.txtを修正する.
  - OpenCVの設定を組み込んでくれるように記述.
  - テキストエディタで編集(メモ帳など)

```
set(comp_srcs Flip.cpp )
set(standalone_srcs FlipComp.cpp)

if (DEFINED OPENRTM_INCLUDE_DIRS)
  string(REGEX REPLACE "-I" ";"
    OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")
  string(REGEX REPLACE " ;" ";"
    OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")
endif (DEFINED OPENRTM_INCLUDE_DIRS)
```

find_package(OpenCV REQUIRED)を追記

```
set(comp_srcs Flip.cpp )
set(standalone_srcs FlipComp.cpp)

find_package(OpenCV REQUIRED)

```

```
add_library(${PROJECT_NAME} ${LIB_TYPE} ${comp_srcs}
  ${comp_headers} ${ALL_IDL_SRCS})
set_target_properties(${PROJECT_NAME} PROPERTIES PREFIX "")
set_source_files_properties(${ALL_IDL_SRCS} PROPERTIES GENERATED 1)
add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES})

add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
  ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES})
```

${OpenCV_LIBS}を追記

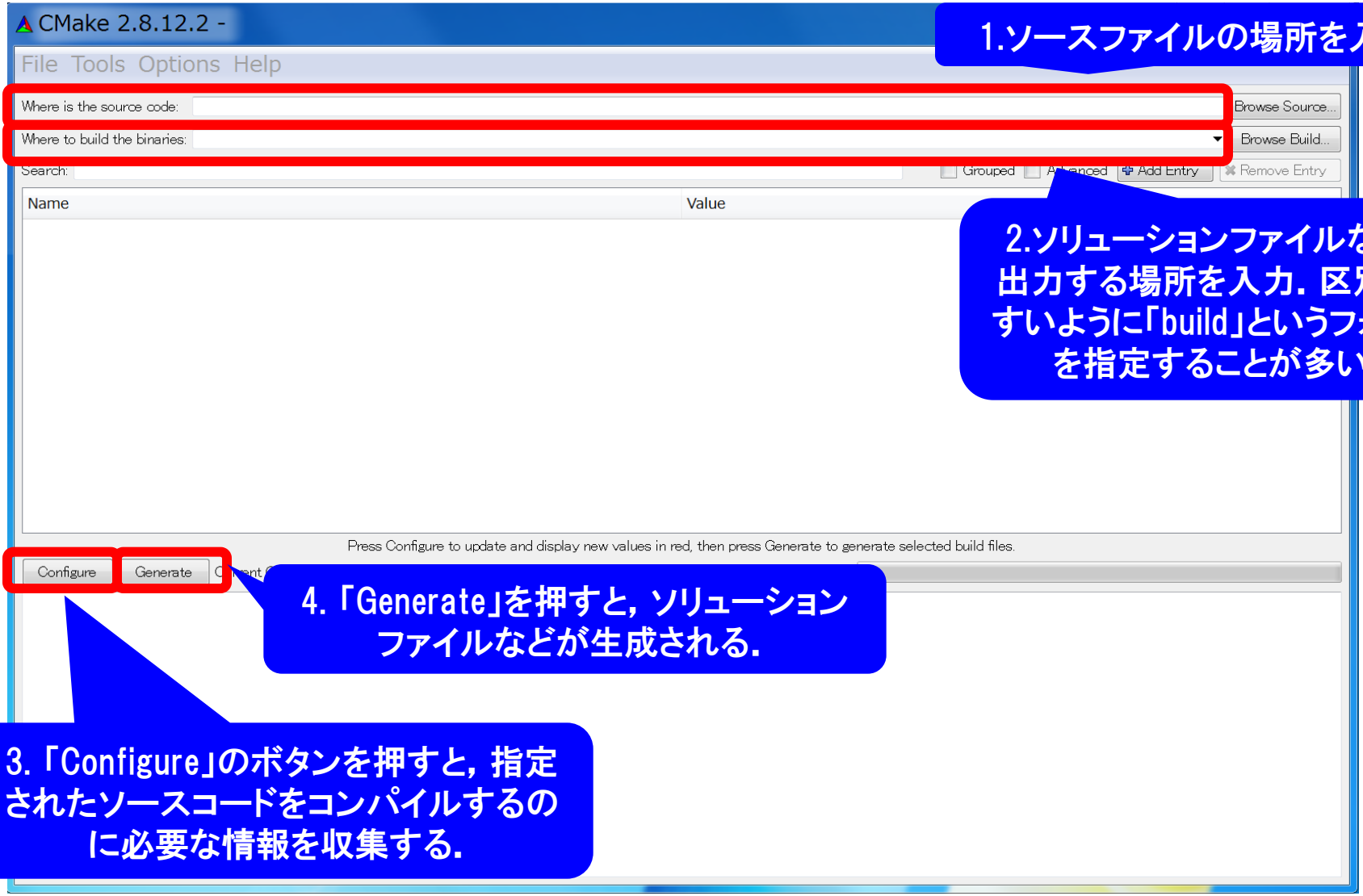
```

add_library(${PROJECT_NAME} ${LIB_TYPE} ${comp_srcs}
  ${comp_headers} ${ALL_IDL_SRCS})
set_target_properties(${PROJECT_NAME} PROPERTIES PREFIX "")
set_source_files_properties(${ALL_IDL_SRCS} PROPERTIES GENERATED 1)
add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES} ${OpenCV_LIBS})

add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
  ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES} ${OpenCV_LIBS})

```

Cmakeの利用

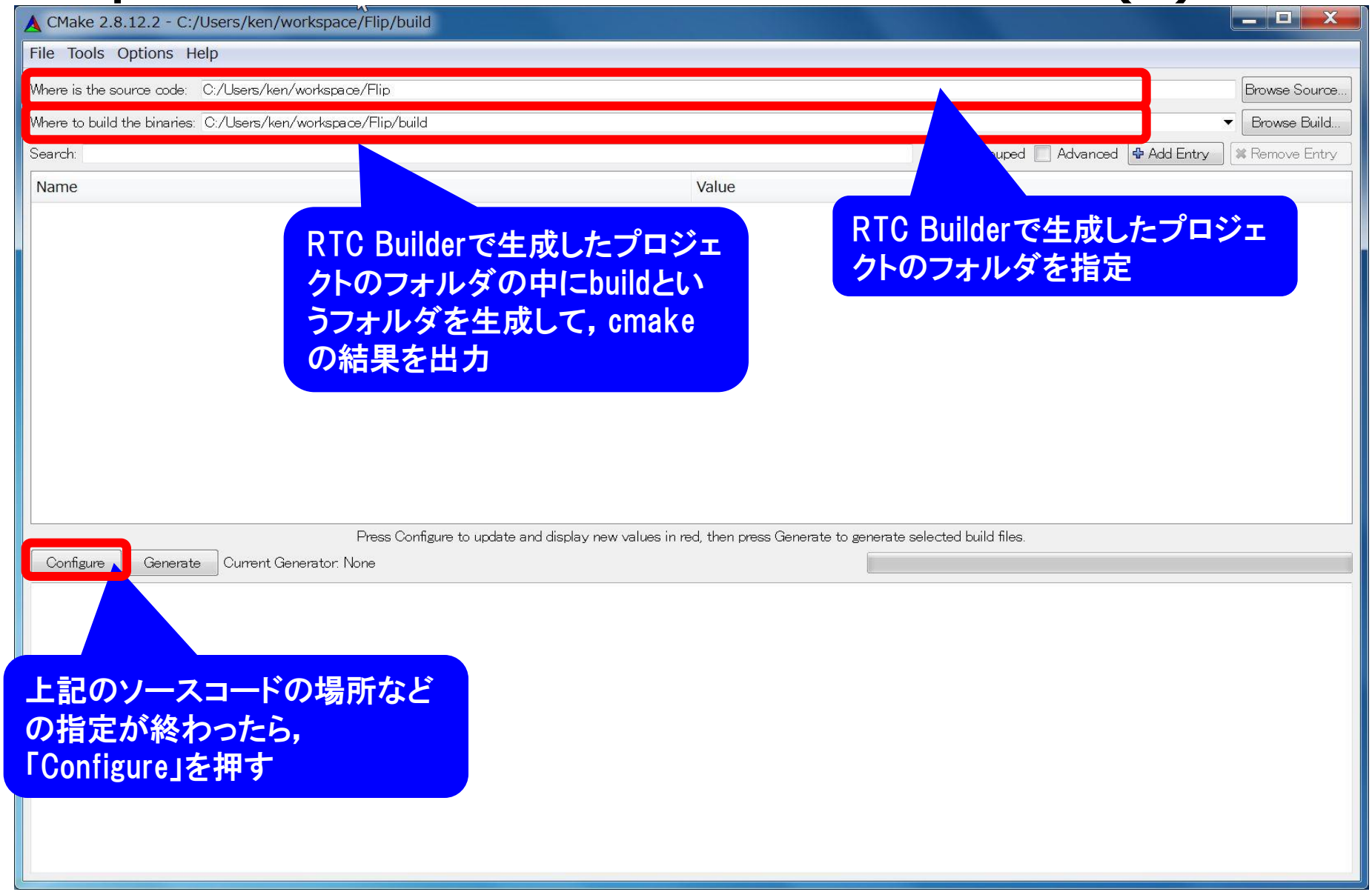
- RTC Builderで出力したファイル群そのものでは，RTCの実行ファイルの生成はできない．
- Cmakeを利用し，ソースファイルのコンパイルに必要な設定が含まれたVisual Studio用のソリューションファイルを生成する．
  - Linuxの場合はソースファイルのコンパイルに必要な設定が含まれたMakefileを生成する．

- Cmakeの起動(Windows 7，Cmake2.8の場合)
  - 「スタート」->「すべてのプログラム」->「Cmake2.8」->「Cmake(Cmake-gui)」
- Ubuntuの場合
  - Dushホームから，Cmakeと入力するとCmake-guiがでてくるので，それを利用．

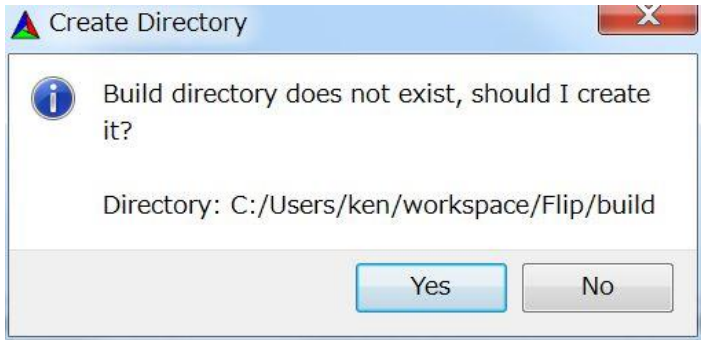
CMakeの起動画面・説明(Windowsの場合)



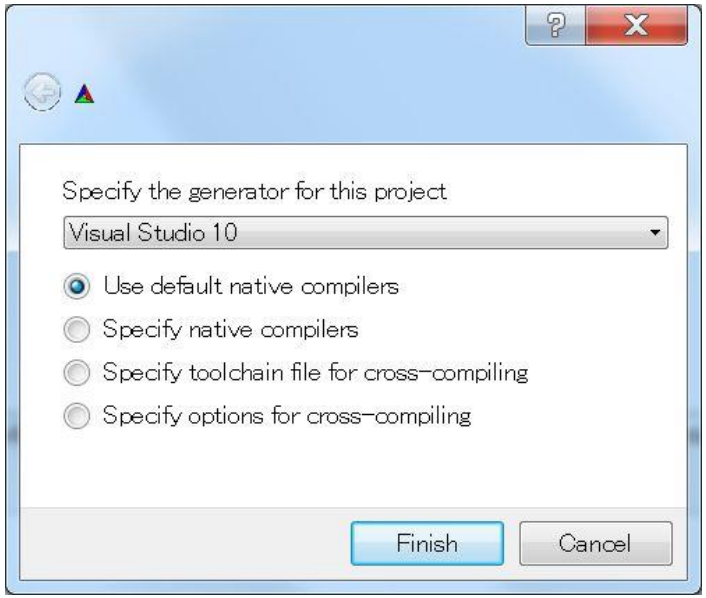
Flipコンポーネントのソースをcmake(1)



Flipコンポーネントのソースをcmake(2)



出力先に指定したbuildのフォルダがない場合，生成する旨が表示される



使用するビルド環境を指定する.

Visual Studioであればそのバージョンを指定.

(Visual Studioのバージョンとの表記の違いに注意)

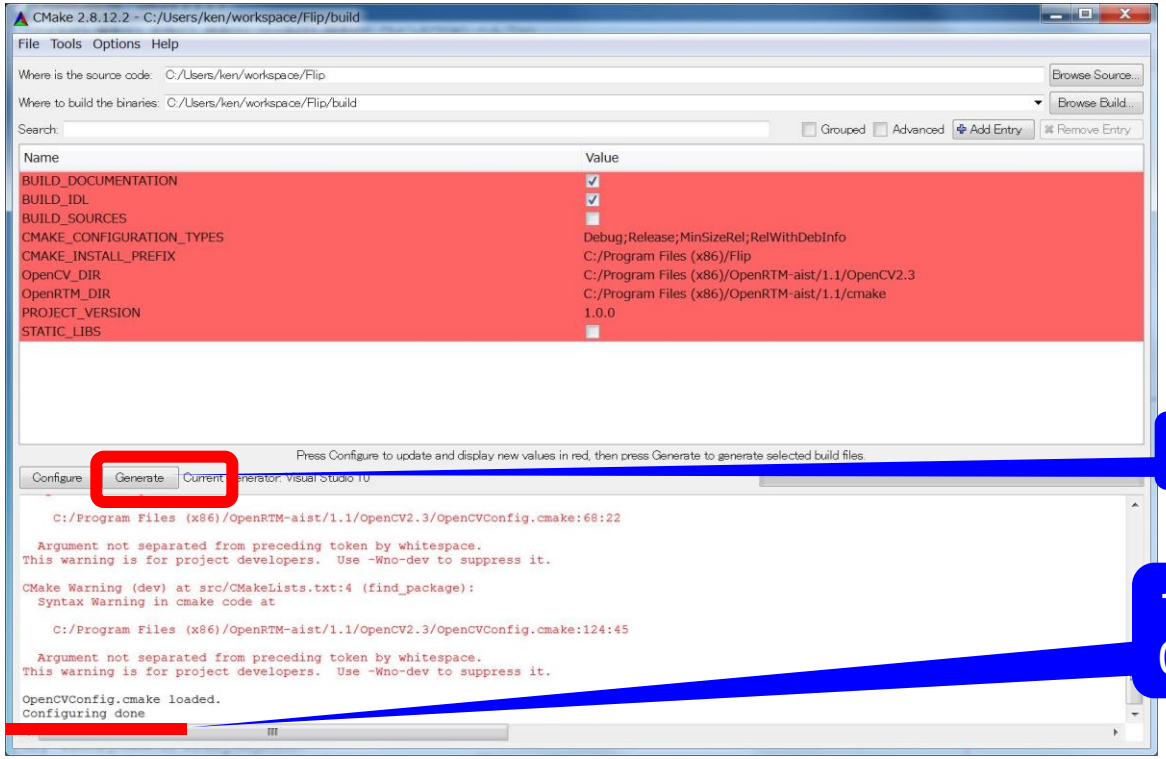
Visual Studio 2010 -> Visual Studio 10

Visual Studio 2012 -> Visual Studio 11

Visual Studio 2013 -> Visual Studio 12

Linuxの場合　Unix Makefiles を指定

Flipコンポーネントのソースをcmake(3)



2.「Generate」をクリック

1.「Configuration Done」と出ていればOK

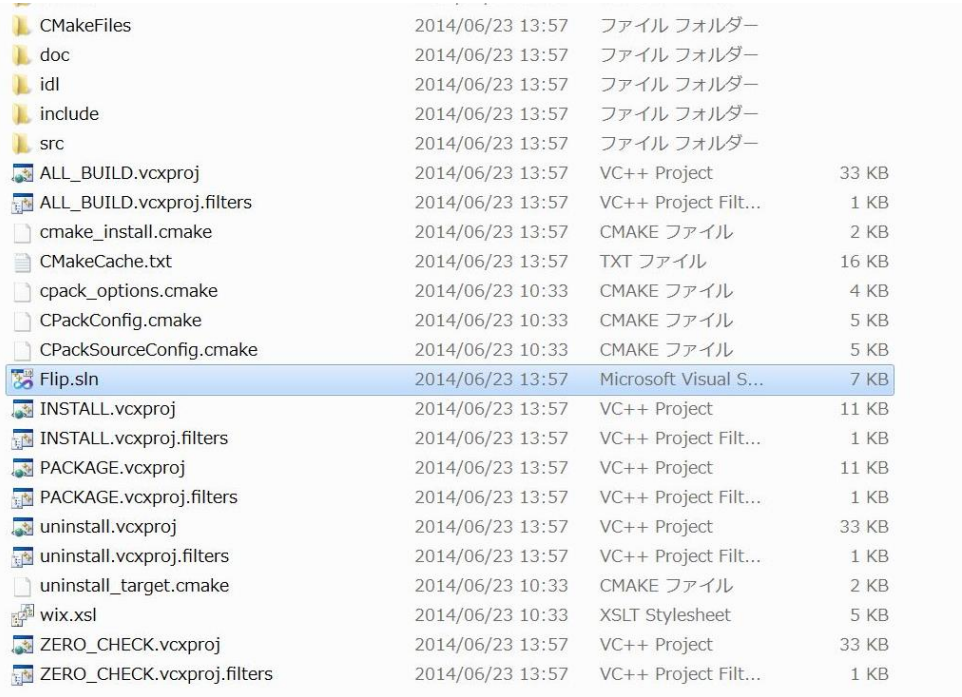
C:/Program Files (x86)/OpenRTM-aist/1.1/OpenCV2.3/OpenCVConfig.cmake:68:22
Argument not separated from preceding token by whitespace.
This warning is for project developers. Use -Wno-dev to suppress it.
CMake Warning (dev) at src/CMakeLists.txt:4 (find_package):
Syntax Warning in cmake code at
C:/Program Files (x86)/OpenRTM-aist/1.1/OpenCV2.3/OpenCVConfig.cmake:124:45
Argument not separated from preceding token by whitespace.
This warning is for project developers. Use -Wno-dev to suppress it.
OpenCVConfig.cmake loaded.
Configuring done
Generating done

3.「Generation Done」と出ればOK

Windowsの場合

Flipコンポーネントのひな形がはき出されたフォルダを<FlipComp_DIR>と表記します．

- <FlipComp_DIR>/buildを開く．
- 「Flip.sln」というソリューションファイルがあるので，そのファイルをダブルクリックして，VisualStudioを起動

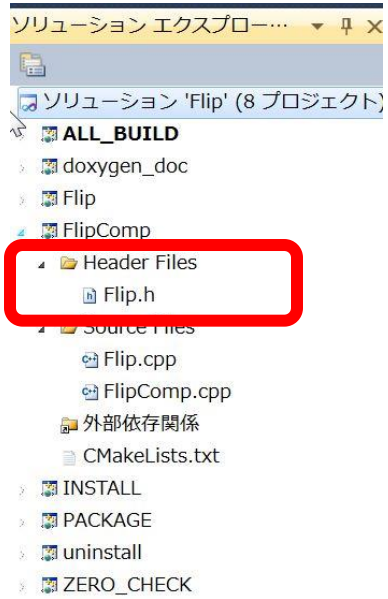


Linuxの場合

- ここでは特に何もしなくてよいです．

ヘッダファイルの修正

- Visual Studioの場合
  - ソリューションエクスプローラにおいて，FlipCompプロジェクトを展開して，Flip.hを開く.
- Linuxの場合
  - <FlipComp_DIR>/include にFlip.hがあるので，これをエディタで開く.



Flip.hの修正箇所

```
#ifndef FLIP_H
#define FLIP_H

#include <rtm/Manager.h>
#include <rtm/DataFlowComponentBase.h>
#include <rtm/CorbaPort.h>
#include <rtm/DataInPort.h>
#include <rtm/DataOutPort.h>
#include <rtm/idl/BasicDataTypeSkel.h>
#include <rtm/idl/ExtendedDataTypesSkel.h>
#include <rtm/idl/InterfaceDataTypesSkel.h>

// Service implementation headers
// <rtc-template block="service_impl_h">

// </rtc-template>

// Service Consumer stub headers
// <rtc-template block="consumer_stub_h">

// </rtc-template>

#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/core/core.hpp>
```

OpenCV用の各種ヘッダファイルを読み込ませる.

```
private:
  // <rtc-template block="private_attribute">

  // </rtc-template>

  // <rtc-template bl

  // </rtc-template>

    cv::Mat m_imagebuf;
    cv::Mat m_flippedbuf;
```

画像処理用変数の定義を追加

ソースファイルの修正

- Visual Studioの場合
  - ソリューションエクスプローラにおいて，FlipCompプロジェクトを展開して，Flip.cppを開く．
- Linuxの場合
  - <FlipComp_DIR>/src にFlip.cppがあるので，これをエディタで開く．

ソースファイルの編集(onActivated)

- onActivatedの場所に，以下のコードを追加

```
108 RTC::ReturnCode_t Flip::onActivated(RTC::UniqueId ec_id)
109 {
110     //Outport用画像サイズの初期化
111     m_flippedImage.width = 0;
112     m_flippedImage.height = 0;
113
114     return RTC::RTC_OK;
115 }
116
```

この部分のコードを追加

ソースファイルの編集(onDeactivated)

- onDeactivatedに以下のコードを追加.

```
122 RTC::ReturnCode_t Flip::onDeactivated(RTC::UniqueId ec_id)
123 {
124     if(!m_imagebuf.empty())
125     {
126         m_imagebuf.release();
127         m_flippedbuf.release();
128
129     }
130     return RTC::RTC_OK;
131 }
```

この部分のコードを追加

ソースファイルの編集(onExecute)

- onExecuteに以下のコードを追加

```
134 RTC::ReturnCode_t Flip::onExecute(RTC::UniqueId ec_id)
135 {
136     //新しいデータのチェック
137     //InPortに入力されたデータが新たに入力されたデータの場合，isNew()の返り値がtrueになる．
138     if( m_originalImageIn.isNew() )
139     {
140         //InPortのデータの読み込み
141         //事前にメモリを確保しなくとも，受信データ長にあわせて自動で確保される．
142         m_originalImageIn.read();
143 
144         //OutPort用の画像サイズの設定，およびメモリの確保
145         if( m_flippedImage.width != m_originalImage.width || m_flippedImage.height != m_originalImage.height )
146         {
147             m_flippedImage.width = m_originalImage.width;
148             m_flippedImage.height = m_originalImage.height;
149 
150             //画像用メモリの確保
151             m_flippedbuf.create(cv::Size(m_originalImage.width, m_originalImage.height), CV_8UC3);
152             m_imagebuf.create(cv::Size(m_originalImage.width, m_originalImage.height), CV_8UC3);
153         }
154 
155         //InPortに入力されたデータをm_imagebufにコピー
156         memcpy(m_imagebuf.data, (void *)&(m_originalImage.pixels[0]), m_originalImage.pixels.length());
157 
158         //InPortからのデータを反転する
159         //m_flipModeの値に応じて反転のモードを変更
160         //0: X軸周り
161         //1: Y軸周り
162         //-1: 両方の軸周り
163         cv::flip(m_imagebuf, m_flippedbuf, m_flipMode);
164 
165         //OutPort用のメモリ確保
166         int length = m_flippedbuf.channels() * m_flippedbuf.cols * m_flippedbuf.rows;
167         m_flippedImage.pixels.length(length);
168 
169         //反転した画像データをOutPortにコピー
170         memcpy((void *)&(m_flippedImage.pixels[0]), m_flippedbuf.data, length);
171 
172         //OutPortから処理結果を出力
173         m_flippedImageOut.write();
174     }
175     return RTC::RTC_OK;
176 }
```

この部分のコードを追加

ソースコードのコンパイル

- Visual Studioの場合
  - ソリューションエクスプローラにおいて，FlipCompプロジェクトで右クリックし，「スタートアッププロジェクトに設定」を選択
  - 「ビルド」→「ソリューションのビルド」を選択
- Linuxの場合
  - <FlipCompDir>/build　においてmakeを実行

何もエラーが出なければコンポーネントの実行ファイルが生成される．エラーが出た場合は，例に見ながらソースファイルとヘッダファイルの記載を確認する．

Flipコンポーネントの実行

- ネーミングサービスを起動する.
- RTSystemEditorを起動する.
- Windowsの場合
  - DirectshowCameraCompおよびCameraViewerCompを起動
  - Visual Studioの場合
    - 以下の画面上部の矢印ボタンを押すと，ビルドと実行が行われる.
    - 少したつとSystem EditorにFlipCompが追加される.
- Linuxの場合
  - Cameraコンポーネント，Viewerコンポーネントを起動
  - <RTC_DIR>/build/srcに移動し，./FlipComp
  - 少したつとSystem EditorにFlipCompが追加される.
- カメラコンポーネント，Flipコンポーネント，ビューワコンポーネントをつないで，「All Activateを行い，動作確認をする．」

まとめ

- Flipコンポーネントの作成を通じて，RTC作成の一連のプロセスを体験.
- 自分でRTCを作る場合は以下のプロセス
  - 作成するRTCの仕様を決定する.
  - 仕様に基づき，RTC BuilderでRTCのひな形を作成
  - CMakeを行い，開発環境に合わせたビルド環境を生成.
  - ヘッダファイル，ソースファイルにコンポーネントの実体を実装.
  - コンポーネントの動作テスト

補足

- 自分のソースコードを公開
  - GitHubなどで公開
  - 公開する場合は，buildフォルダは削除した状態で公開すること．
    - 特定の環境情報を含んだビルド環境であるため．
  - マニュアルの整備をきちんと行うこと！

- 人の作成したRTCを利用
  - GitHubなどで公開されているRTCのソースコードの多くは，CMakeを行いビルド環境を構築すれば，ビルドし，実行してみることが可能．
    - 必要なライブラリがある場合は適宜その環境構築が必要．
  - マニュアルをよく読むこと！
  - 学習の意味で，ソースコードにも目を通してみる．！
    - ソースコードがRTミドルウェア学習のための一番の教科書です．

RTミドルウェアサマーキャンプのご案内

- 8月4日～8日まで，茨城県つくば市の産業技術総合研究所において開催
- 所属の異なるメンバーでグループを構成し，協力しながら期間内でグループのコンセプトにあったシステムを構築.
- RTミドルウェアに精通した講師陣のサポートをいつでも受けれる！
- 普段の講習会では時間の都合で困難な，RTミドルウェアに関連する様々な講演を聴講できる！
- 日本全国に知人ができる！
  - 昨年度の参加者はFacebookなどで今もつながっています！

RTミドルウェアコンテストのご案内

- SICE SI（計測自動制御学会システムインテグレーション部門講演会）のセッションとして開催
  - エントリー〆切：8月22日
  - 講演原稿〆切：9月26日
  - ソフトウエア登録：12月初旬ごろ
  - 発表・授賞式：12月15日（月）～17日（水）
    於：東京ビッグサイト

- 詳細はWebページ： openrtm.org
  - コミュニティー→イベントをご覧ください

RTミドルウェアを利用したシステム開発を行っていく一つのモチベーションとしてこの機会をご利用ください！