

第3部：プログラミング演習

名城大学

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)

```

1 set(comp_srcs Flip.cpp )←
2 set(standalone_srcs FlipComp.cpp)←
3 ←
4 if (DEFINED OPENRTM_INCLUDE_DIRS)←
5     string(REGEX REPLACE "-I";" "←
6         OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")←
7     string(REGEX REPLACE ";" " ";←
8         OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")←
9 endif (DEFINED OPENRTM_INCLUDE_DIRS)←

```

find_package(OpenCV REQUIRED)を追記

```

1 set(comp_srcs Flip.cpp )←
2 set(standalone_srcs FlipComp.cpp)←
3 ←
4 find_package(OpenCV REQUIRED)←
5 ←

```

```

41 add_library(${PROJECT_NAME} ${LIB_TYPE} ${comp_srcs}←
42     ${comp_headers} ${ALL_IDL_SRCS})←
43 set_target_properties(${PROJECT_NAME} PROPERTIES PREFIX "")←
44 set_source_files_properties(${ALL_IDL_SRCS} PROPERTIES GENERATED 1)←
45 add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)←
46 target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES})←
47 ←
48 add_executable(${PROJECT_NAME}Comp ${standalone_srcs}←
49     ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})←
50 target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES})←

```

`\${OpenCV_LIBS}`を追記

```

40 ←
41 add_library(${PROJECT_NAME} ${LIB_TYPE} ${comp_srcs}←
42     ${comp_headers} ${ALL_IDL_SRCS})←
43 set_target_properties(${PROJECT_NAME} PROPERTIES PREFIX "")←
44 set_source_files_properties(${ALL_IDL_SRCS} PROPERTIES GENERATED 1)←
45 add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)←
46 target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES} `${OpenCV_LIBS}`)←
47 ←
48 add_executable(${PROJECT_NAME}Comp ${standalone_srcs}←
49     ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})←
50 target_link_libraries(${PROJECT_NAME}Comp `${OpenCV_LIBS}`)←
51 ←

```

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)

The screenshot shows the CMake 2.8.12.2 GUI. The title bar reads 'CMake 2.8.12.2 -'. The menu bar includes 'File Tools Options Help'. The main area contains two input fields: 'Where is the source code:' and 'Where to build the binaries:'. Below these is a search bar and a table with columns 'Name' and 'Value'. At the bottom, there are 'Configure' and 'Generate' buttons. A status bar at the bottom reads 'Press Configure to update and display new values in red, then press Generate to generate selected build files.'

1. ソースファイルの場所を入力

2. ソリューションファイルなどを出力する場所を入力. 区別しやすいように「build」というフォルダを指定することが多い.

3. 「Configure」のボタンを押すと, 指定されたソースコードをコンパイルするのに必要な情報を収集する.

4. 「Generate」を押すと, ソリューションファイルなどが生成される.

FlipコンポーネントのソースをCMake(1)

CMake 2.8.12.2 - C:/Users/ken/workspace/Flip/build

File Tools Options Help

Where is the source code: C:/Users/ken/workspace/Flip

Where to build the binaries: C:/Users/ken/workspace/Flip/build

Search: Grouped Advanced

Name	Value
------	-------

Press Configure to update and display new values in red, then press Generate to generate selected build files.

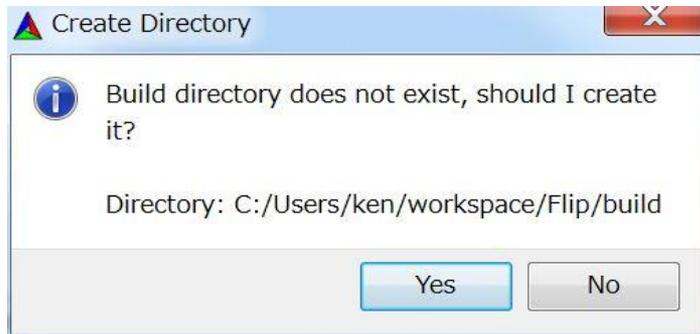
Current Generator: None

RTC Builderで生成したプロジェクトのフォルダの中にbuildというフォルダを生成して, cmakeの結果を出力

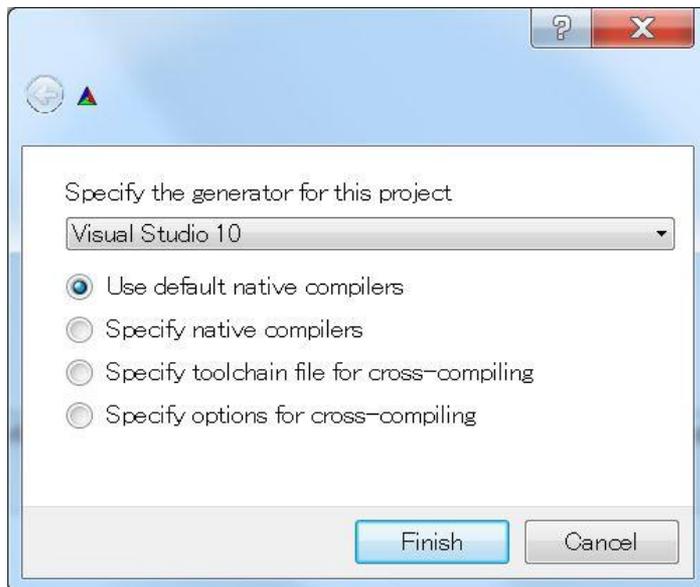
RTC Builderで生成したプロジェクトのフォルダを指定

上記のソースコードの場所などの指定が終わったら, 「Configure」を押す

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)

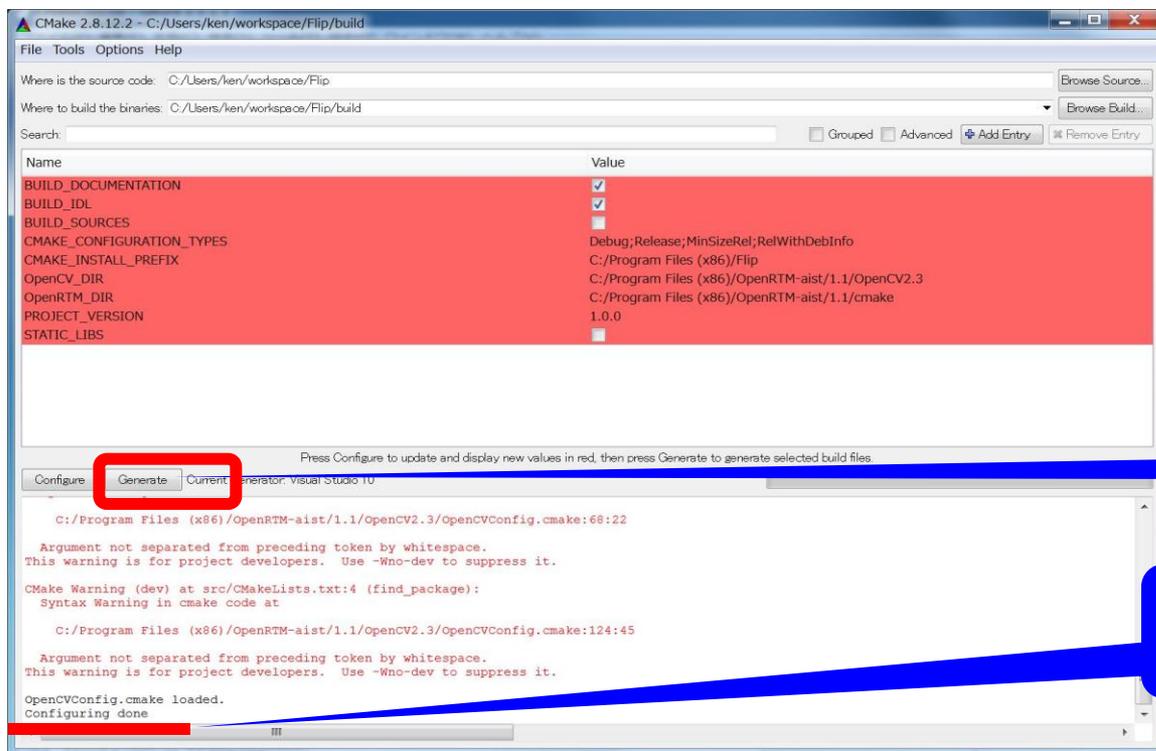
Visual Studio 2010 -> Visual Studio 10

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

3. 「Generation Done」と出れば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
```

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

CMakeFiles	2014/06/23 13:57	ファイル フォルダー	
doc	2014/06/23 13:57	ファイル フォルダー	
idl	2014/06/23 13:57	ファイル フォルダー	
include	2014/06/23 13:57	ファイル フォルダー	
src	2014/06/23 13:57	ファイル フォルダー	
ALL_BUILD.vcxproj	2014/06/23 13:57	VC++ Project	33 KB
ALL_BUILD.vcxproj.filters	2014/06/23 13:57	VC++ Project Filt...	1 KB
cmake_install.cmake	2014/06/23 13:57	CMAKE ファイル	2 KB
CMakeCache.txt	2014/06/23 13:57	TXT ファイル	16 KB
cpack_options.cmake	2014/06/23 10:33	CMAKE ファイル	4 KB
CPackConfig.cmake	2014/06/23 10:33	CMAKE ファイル	5 KB
CPackSourceConfig.cmake	2014/06/23 10:33	CMAKE ファイル	5 KB
Flip.sln	2014/06/23 13:57	Microsoft Visual S...	7 KB
INSTALL.vcxproj	2014/06/23 13:57	VC++ Project	11 KB
INSTALL.vcxproj.filters	2014/06/23 13:57	VC++ Project Filt...	1 KB
PACKAGE.vcxproj	2014/06/23 13:57	VC++ Project	11 KB
PACKAGE.vcxproj.filters	2014/06/23 13:57	VC++ Project Filt...	1 KB
uninstall.vcxproj	2014/06/23 13:57	VC++ Project	33 KB
uninstall.vcxproj.filters	2014/06/23 13:57	VC++ Project Filt...	1 KB
uninstall_target.cmake	2014/06/23 10:33	CMAKE ファイル	2 KB
wix.xsl	2014/06/23 10:33	XSLT Stylesheet	5 KB
ZERO_CHECK.vcxproj	2014/06/23 13:57	VC++ Project	33 KB
ZERO_CHECK.vcxproj.filters	2014/06/23 13:57	VC++ Project Filt...	1 KB

Linuxの場合

- ここでは特に何もしなくてよいです。

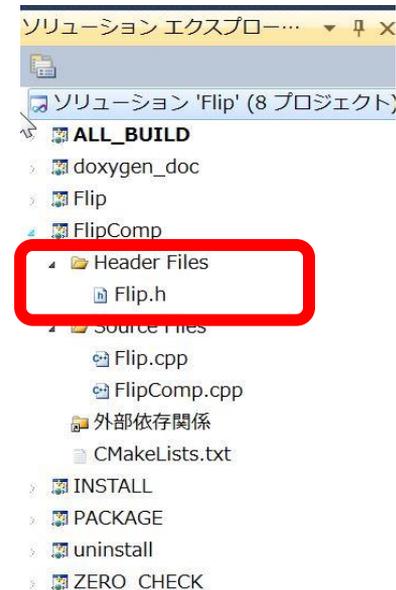
ヘッダファイルの修正

■ 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

```

10 #ifndef FLIP_H
11 #define FLIP_H
12
13 #include <rtm/Manager.h>
14 #include <rtm/DataFlowComponentBase.h>
15 #include <rtm/CorbaPort.h>
16 #include <rtm/DataInPort.h>
17 #include <rtm/DataOutPort.h>
18 #include <rtm/idl/BasicDataTypesSkel.h>
19 #include <rtm/idl/ExtendedDataTypesSkel.h>
20 #include <rtm/idl/InterfaceDataTypesSkel.h>
21
22 // Service implementation headers
23 // <rtc-template block="service_impl_h">
24
25 // </rtc-template>
26
27 // Service Consumer stub headers
28 // <rtc-template block="consumer_stub_h">
29
30 // </rtc-template>
31
32 #include<opencv2/imgproc/imgproc.hpp>
33 #include<opencv2/highgui/highgui.hpp>
34 #include<opencv2/core/core.hpp>

```

OpenCV用の各種ヘッダファイルを読み込ませる。

```

273 private:←
274 // <rtc-template block="private_attribute">←
275 ←
276 // </rtc-template>←
277 ←
278 // <rtc-template bl
279 ←
280 // </rtc-template>←
281 ←
282 ^ cv::Mat m_imagebuf;←
283 ^ 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     //Output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     return RTC::RTC_OK;
130 }
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  }
177 }
```

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
 - ソフトウェア登録: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→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！**