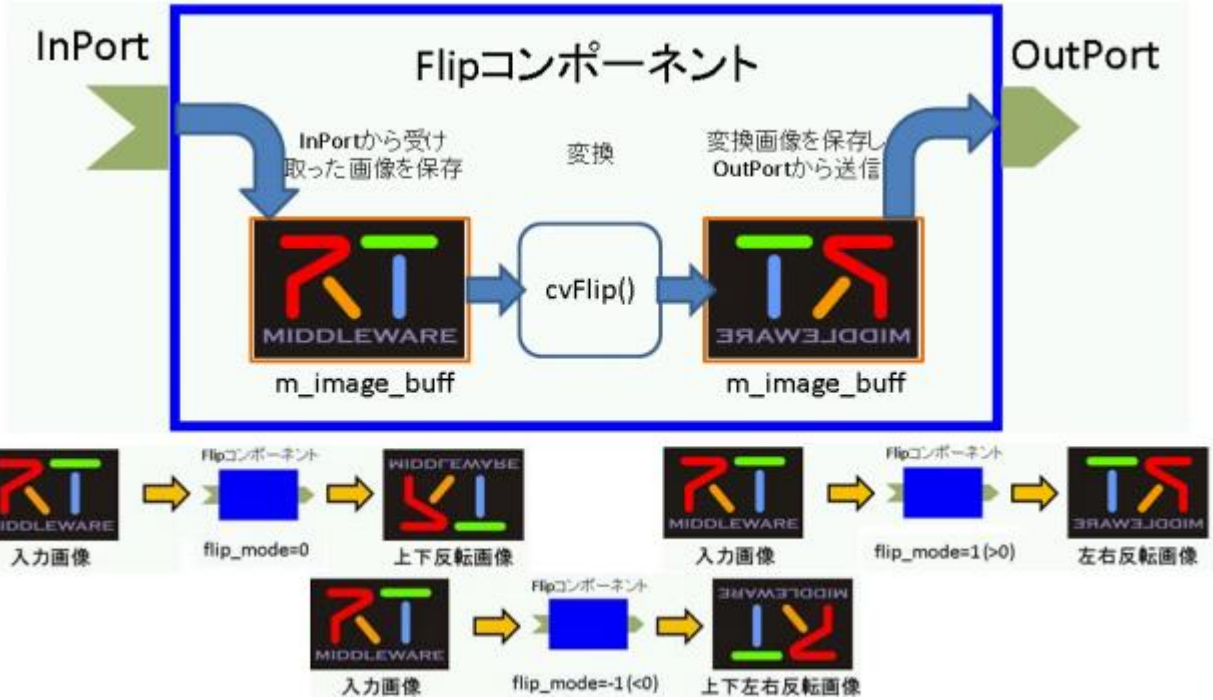
第3部
プログラミング実習

宮本 信彦
国立研究開発法人産業技術総合研究所
ロボットイノベーション研究センター
ロボットソフトウェアプラットフォーム研究チーム

実習内容

- 画像の反転を行うコンポーネントの作成
  - InPortで受信した画像データを処理してOutPortから出力
    - データポートの使用方法を習得
  - コンフィギュレーションパラメータにより反転する方向を設定
    - コンフィギュレーションパラメータの使用方法を習得
  - RT System Editorにより他のRTCと接続、RTCをアクティブ化
    - RT System Editorの使い方を習得



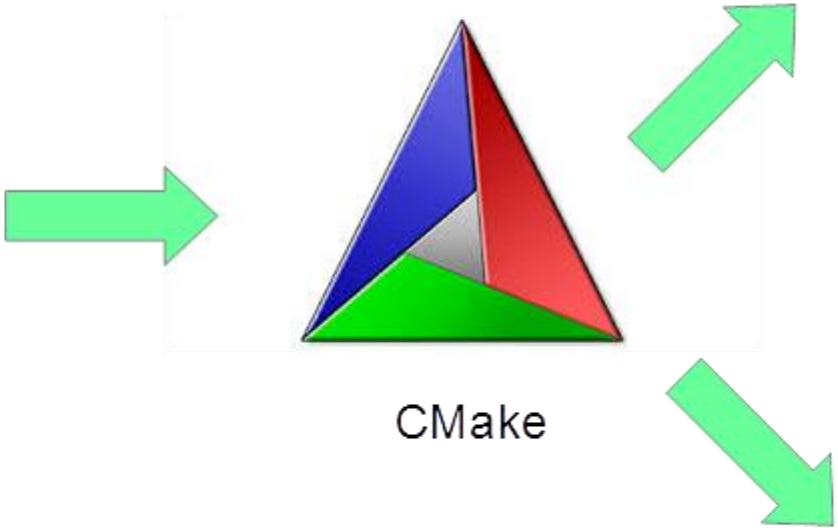
全体の手順

- RTC Builderによるスケルトンコードの作成
  - 作成済み
- ビルドに必要な各種ファイルを生成
  - CMakeLists.txtの編集
  - Cmakeにより各種ファイル生成
- ソースコードの編集
  - Flip.hの編集
  - Flip.cppの編集
- ビルド
- 動作確認

CMake

- ビルドに必要な各種ファイルを生成
  - CMakeLists.txtに設定を記述
    - RTC Builderでスケルトンコードを作成した時にCMakeLists.txtも生成されている

CMakeLists.txt



CMake

Visual Studio
(ソリューションファイル、
プロジェクトファイル等)

Makefile

CMakeLists.txtの編集

- OpenCVを利用するためにCMakeLists.txtを修正する
  - worksapce¥FlipのsrcフォルダのCMakeLists.txtをメモ帳などで開いて編集する

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

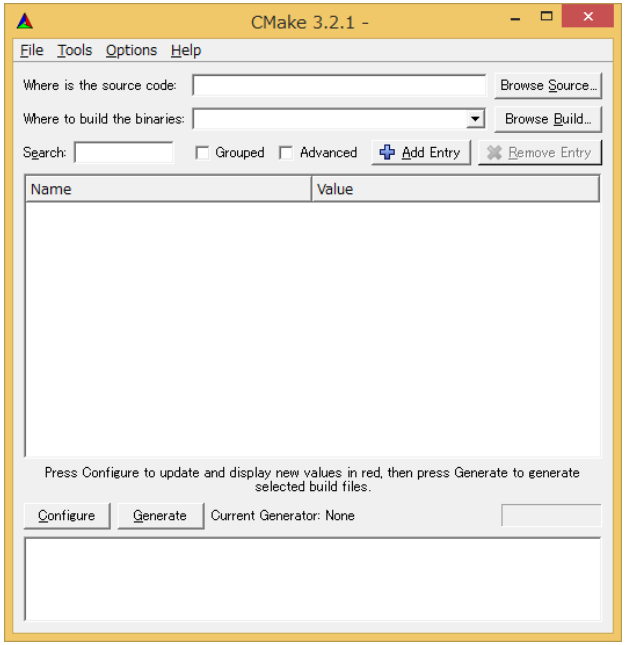
find_package(OpenCV REQUIRED)
を追加

```
47 endif(NOT TARGET ALL_IDL_TGT)
48 add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
49 target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES} ${OpenCV_LIBS})
50
51 add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
52   ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
53 target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES} ${OpenCV_LIBS})
54
```

target_link_librariesの引数に${OpenCV_LIBS}を追加
※2箇所あるので注意

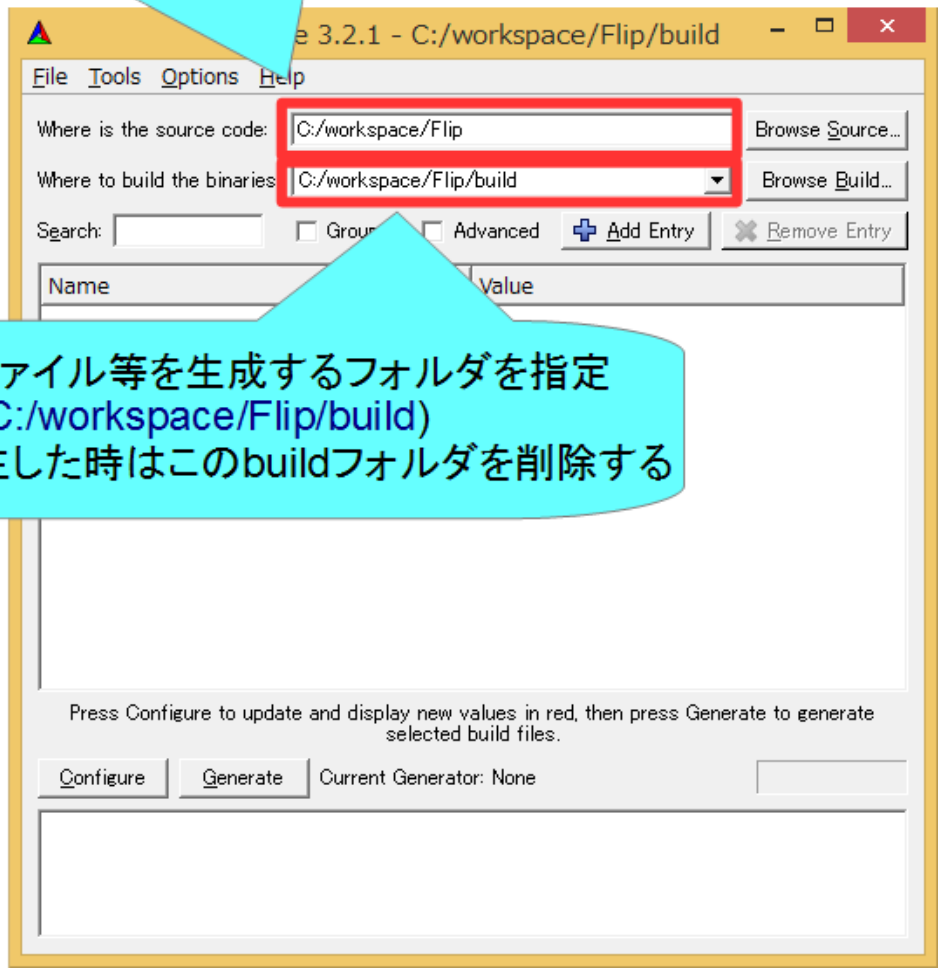
ビルドに必要なファイルの生成

- CMakeを使用する
  - Windows 7
    - 「スタート」→「すべてのプログラム」→「CMake 3.5.2」→「CMake (cmake-gui)」
  - Windows 8.1
    - 「スタート」→「アプリビュー(右下矢印)」→「CMake 3.5.2」→「CMake (cmake-gui)」
  - Ubuntu
    - コマンドで「cmake-gui」を入力



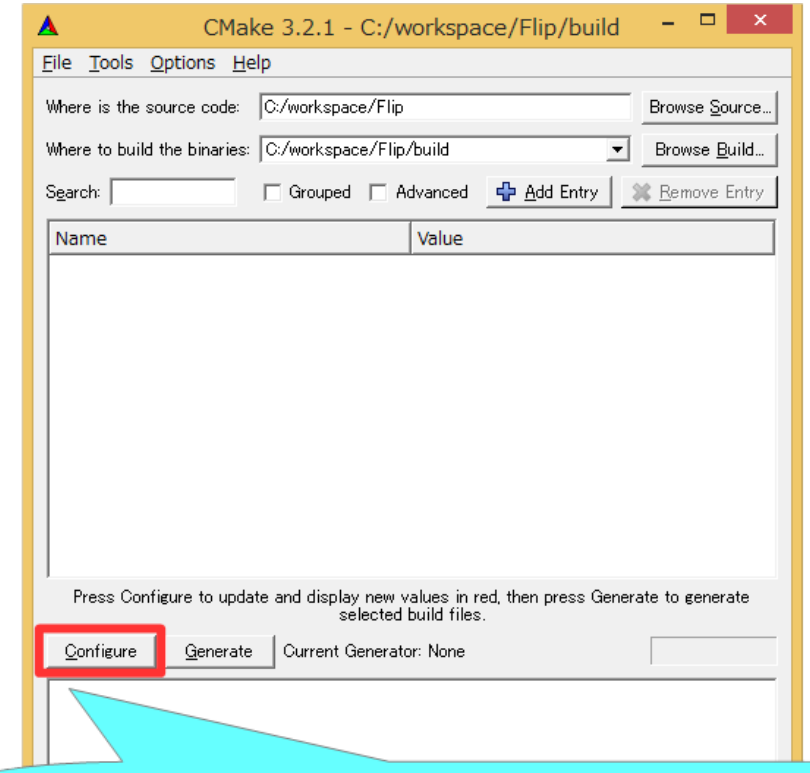
ビルドに必要なファイルの生成

RTC Builderで生成したプロジェクトのフォルダを指定
(例：C:/workspace/Flip)

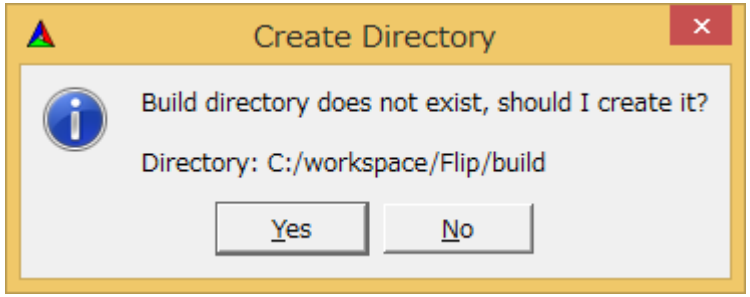


ソリューションファイル等を生成するフォルダを指定
(例：C:/workspace/Flip/build)
何かトラブルが発生した時はこのbuildフォルダを削除する

ビルドに必要なファイルの生成



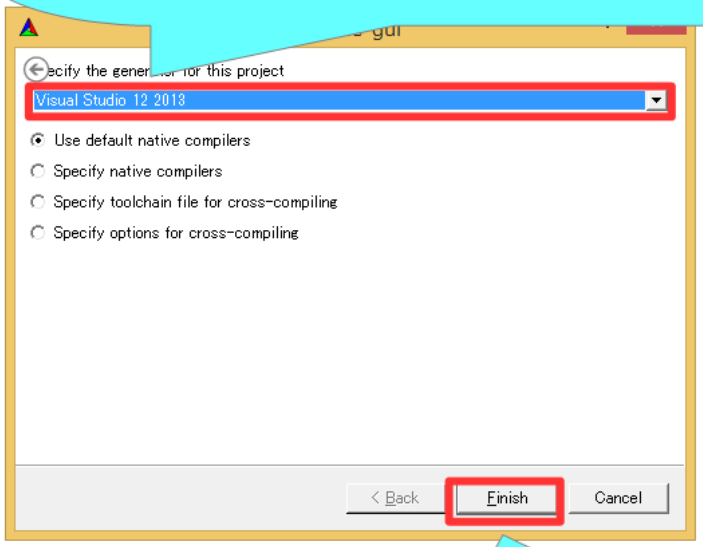
Configureボタンを押す
コンパイルに必要な情報の収集(必要なライブラリの検出など)を行う



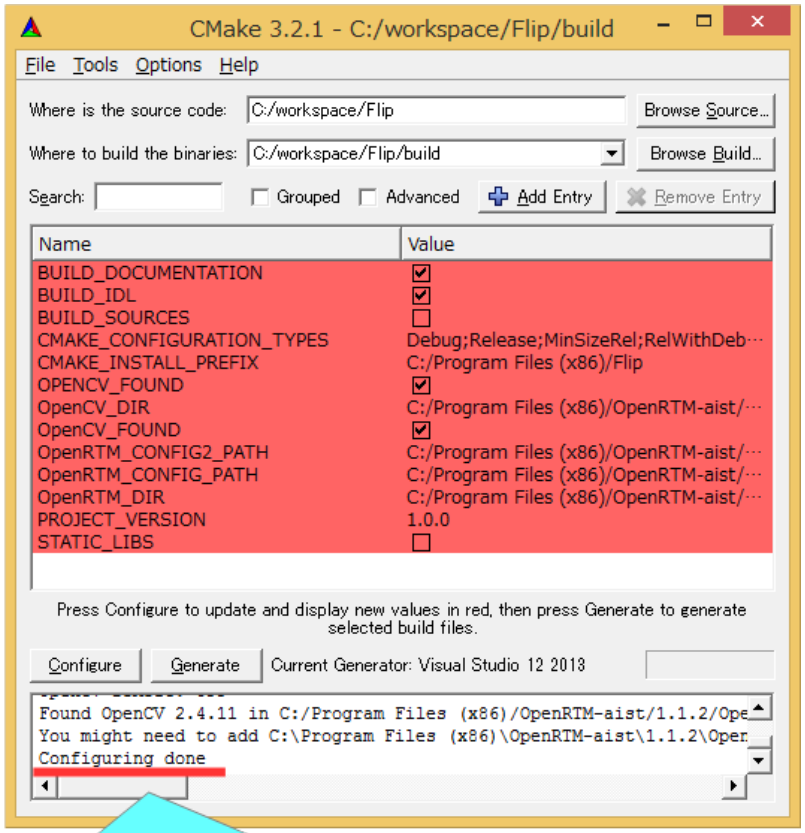
buildフォルダが存在しない場合は
作成するかどうかきかれるため「Yes」を選択

ビルドに必要なファイルの生成

ビルド環境の設定
Visual Studio 2010 32bit → Visual Studio 10 2010
Visual Studio 2012 32bit → Visual Studio 11 2012
Visual Studio 2012 64bit → Visual Studio 11 2012 Win 64
Visual Studio 2013 32bit → Visual Studio 12 2013
Visual Studio 2013 64bit → Visual Studio 12 2013 Win 64
Code::Blocks → CodeBlocks -Unix Makefiles

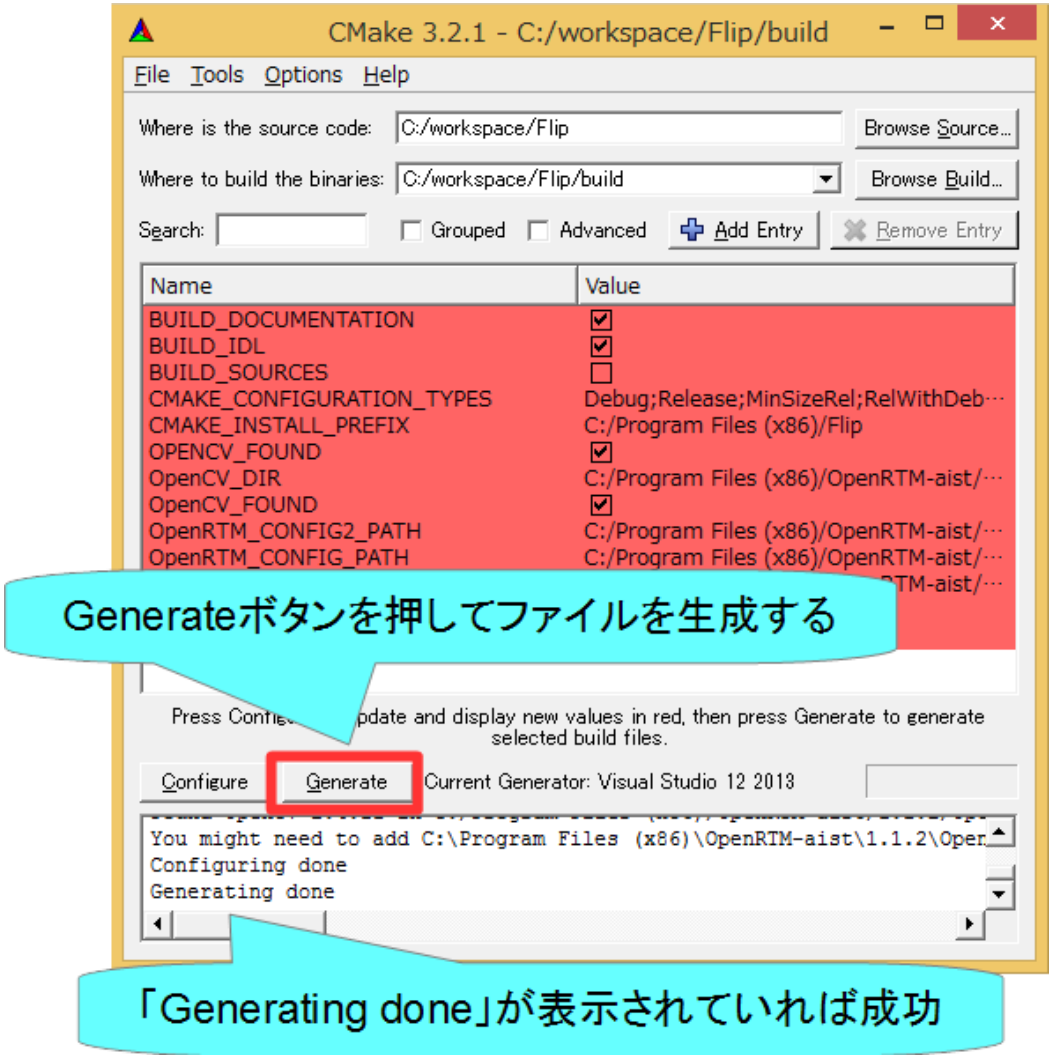


設定後、Finishボタンを押す



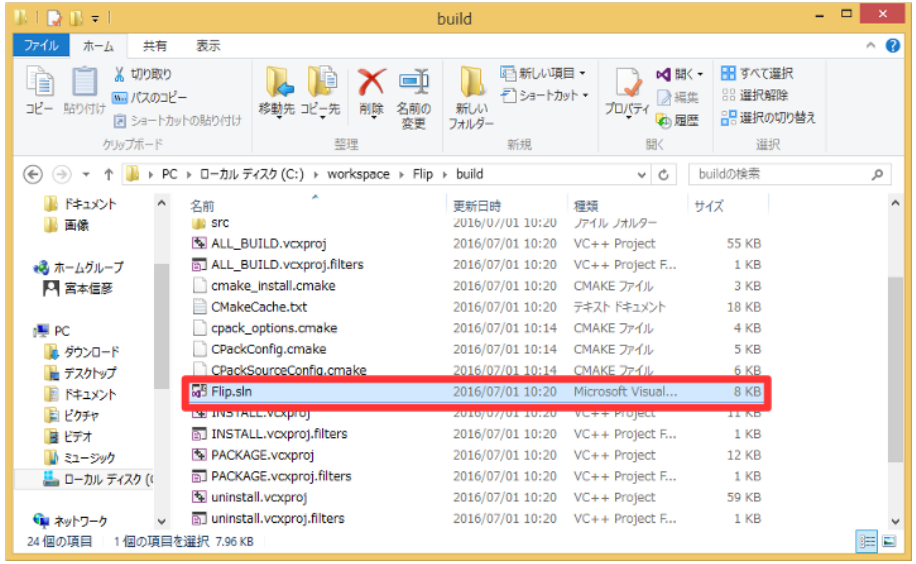
「Configure done」が表示されていれば成功

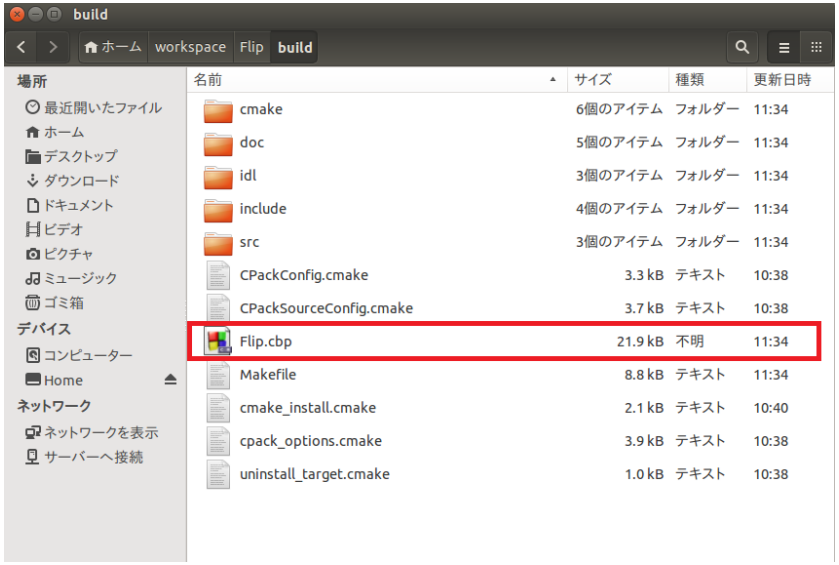
ビルドに必要なファイルの生成



ソースコードの編集

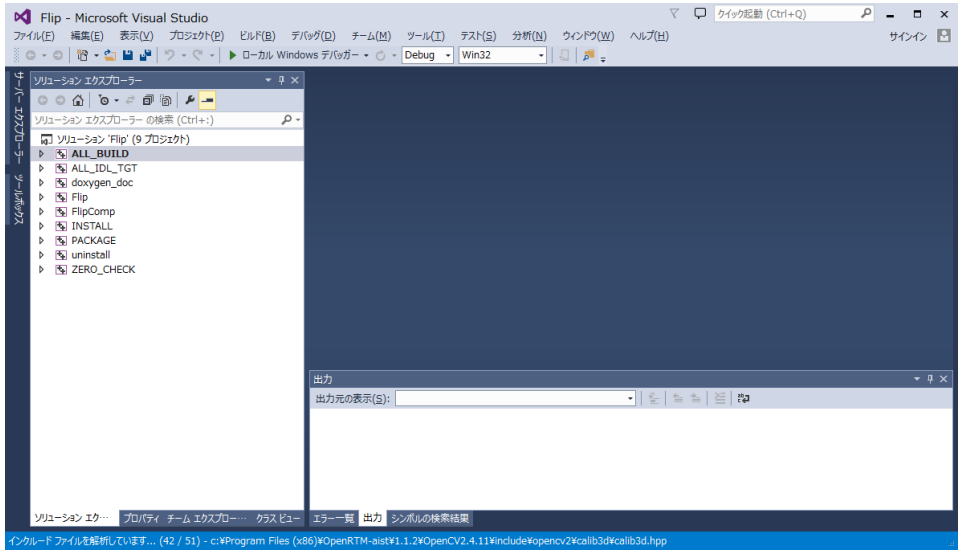
- Windows
  - buildフォルダの「Flip.sln」をダブルクリックして開く
- Ubuntu
  - buildフォルダの「Flip.cbp」をダブルクリックして開く

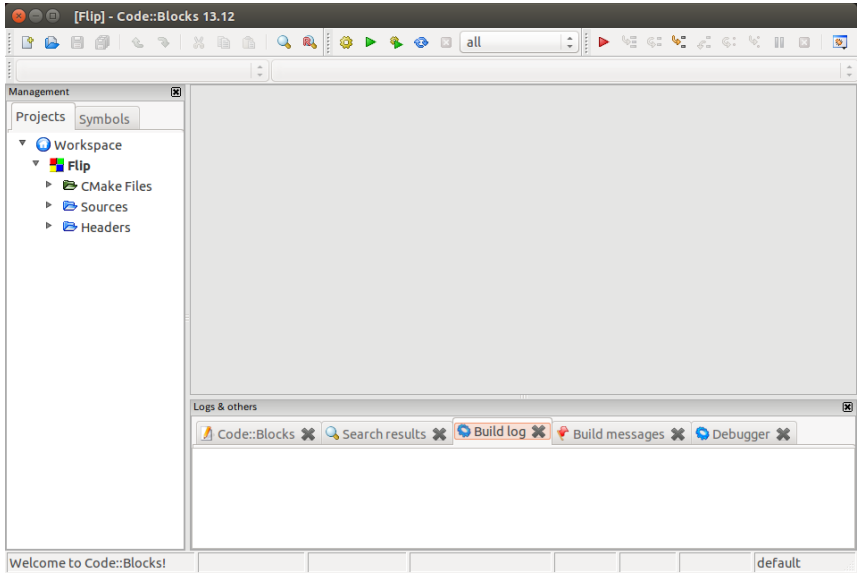




ソースコードの編集

- Windows
  - Visual Studioが起動
- Ubuntu
  - Code::Blocksが起動

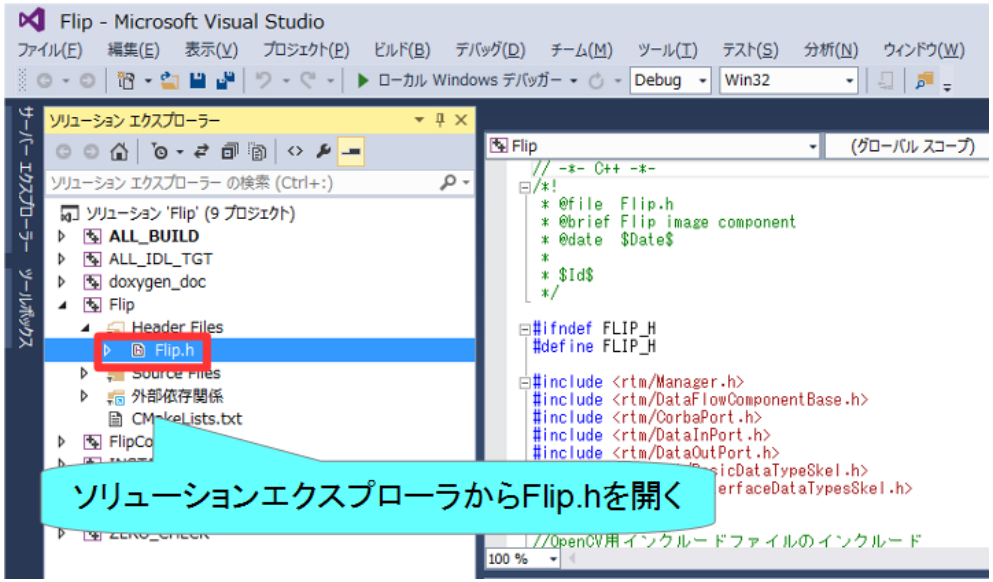




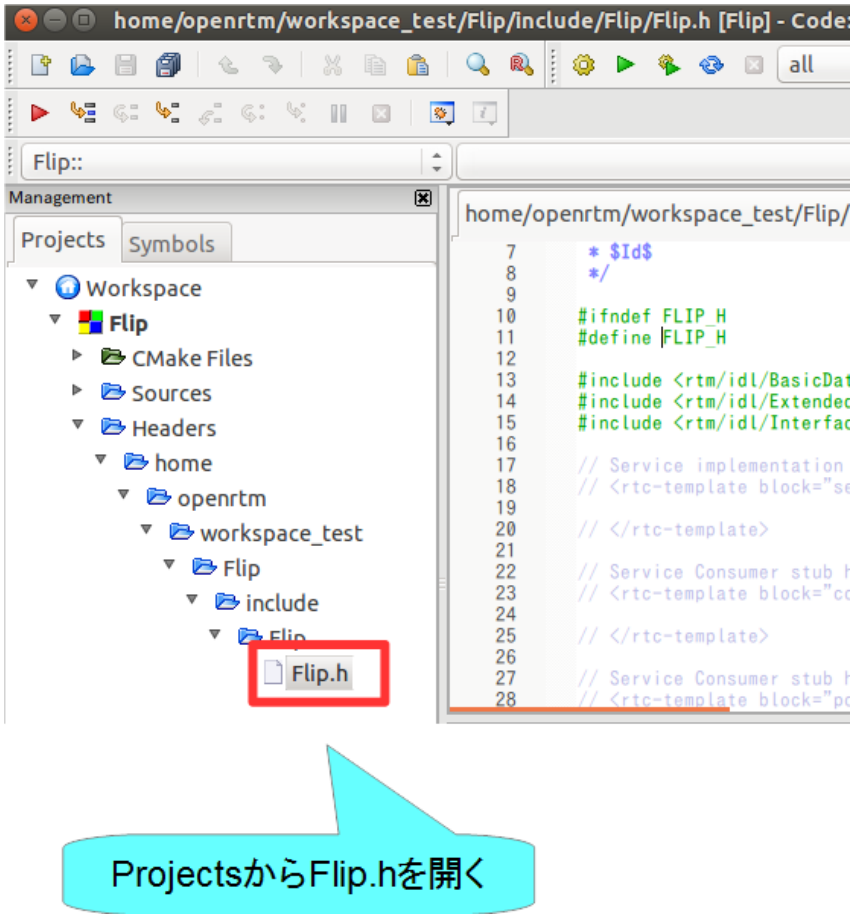
ソースコードの編集

- Flip.hの編集

Visual Studio



Code::Blocks



ソースコードの編集

- Flip.hの編集

```
19  #include <rtm/idl/InterfaceDataTypesSkel.h>
20
21
22  //OpenCV用インクルードファイルのインクルード
23  #include <cv.h>
24
25
26  // lementation headers
```

OpenCVのヘッダーファイルをインクルードする
#include <cv.h>

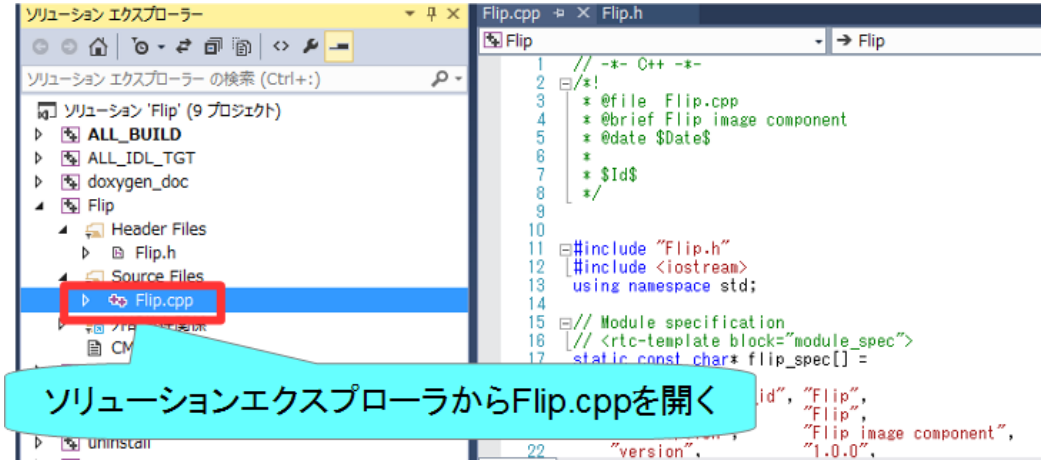
```
272  private:
273   // <rtc-template block="private_attribute">
274
275   // </rtc-template>
276
277   // <rtc-template block="private_operation">
278
279   // </rtc-template>
280       cv::Mat m_imageBuff;
281       cv::Mat m_flipImageBuff;
282
283  };
```

変数の宣言
cv::Mat m_imageBuff;
cv::Mat m_flipImageBuff;

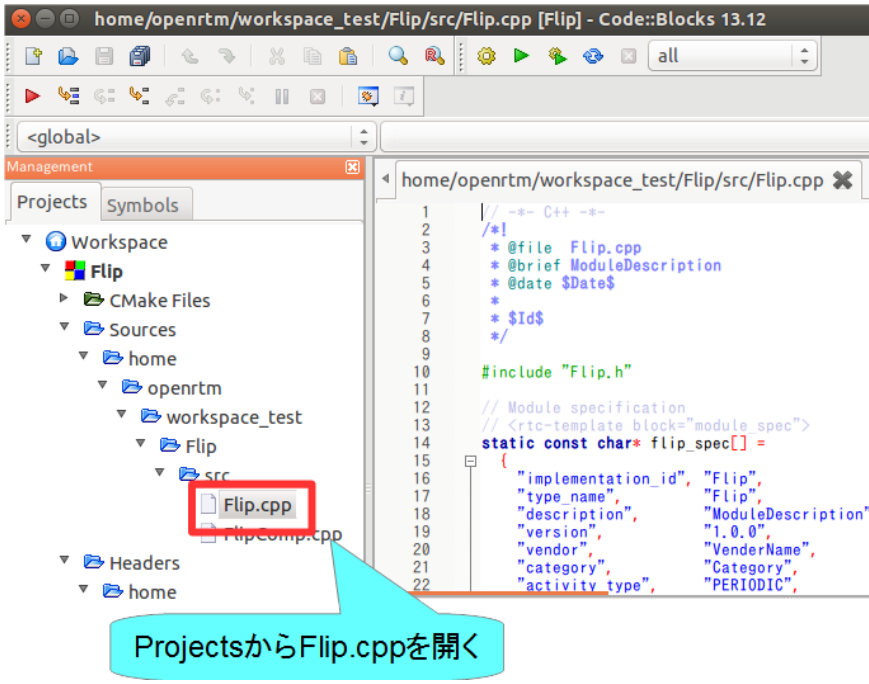
ソースコードの編集

- Flip.cppの編集

Visual Studio



Code::Blocks



ソースコードの編集

- Flip.cppの編集

```
110 RTC::ReturnCode_t Flip::onActivated(RTC::UniqueId ec_id)
111 {
112
113     // OutPortの画面サイズの初期化
114     m_flippedImage.width = 0;
115     m_flippedImage.height = 0;
116
117     return RTC::RTC_OK;
118 }
```

onActivateに追加

```
121 RTC::ReturnCode_t Flip::onDeactivated(RTC::UniqueId ec_id)
122 {
123     if (!m_imageBuff.empty())
124     {
125         // 画像用メモリの解放
126         m_imageBuff.release();
127         m_flipImageBuff.release();
128     }
129
130     return RTC::RTC_OK;
131 }
```

onDeactivateに追加

ソースコードの編集

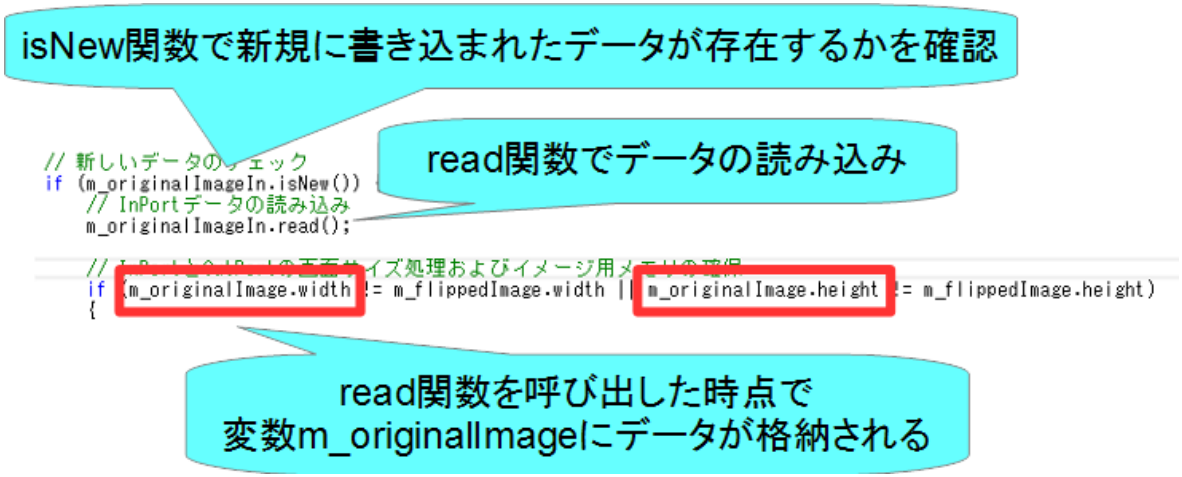
- Flip.cppの編集

```
RTC::ReturnCode_t Flip::onExecute(RTC::UniqueId ec_id)
{
    // 新しいデータのチェック
    if (m_originalImageIn.isNew()) {
        // InPortデータの読み込み
        m_originalImageIn.read();

        // InPortとOutPortの画面サイズ処理およびイメージ用メモリの確保
        if (m_originalImage.width != m_flippedImage.width || m_originalImage.height != m_flippedImage.height)
        {
            m_flippedImage.width = m_originalImage.width;
            m_flippedImage.height = m_originalImage.height;

            m_imageBuff.create(cv::Size(m_originalImage.width, m_originalImage.height), CV_8UC3);
            m_flipImageBuff.create(cv::Size(m_originalImage.width, m_originalImage.height), CV_8UC3);


        }

        // InPortの画像データをm_imageBuffにコピー
        memcpy(m_imageBuff.data, (void *)&(m_originalImage.pixels[0]), m_originalImage.pixels.length());

        // InPortからの画像データを反転する。 m_flipMode 0: X軸周り, 1: Y軸周り, -1: 両方の軸周り
        cv::flip(m_imageBuff, m_flipImageBuff, m_flipMode);

        // 画像データのサイズ取得
        int len = m_flipImageBuff.channels() * m_flipImageBuff.cols * m_flipImageBuff.rows;
        m_flippedImage.pixels.length(len);

        // 反転した画像データをOutPortにコピー
        memcpy((void *)&(m_flippedImage.pixels[0]), m_flipImageBuff.data, len);

        // 反転した画像データをOutPortから出力する。
        m_flippedImageOut.write();
    }

    return RTC::RTC_OK;
}
```

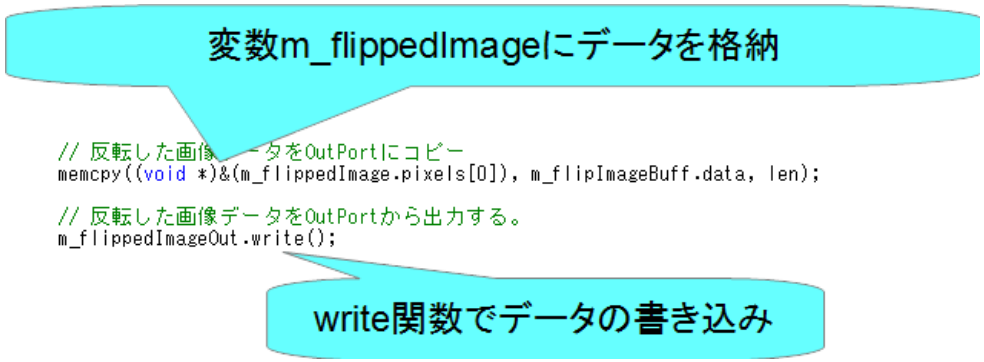
onExecuteに追加

ソースコードの編集

- データを読み込む手順

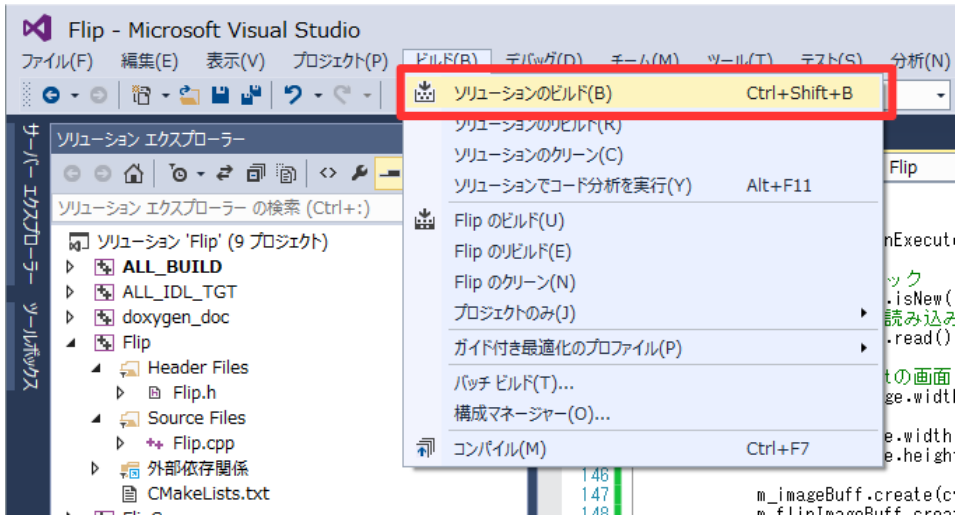


- データを書き込む手順

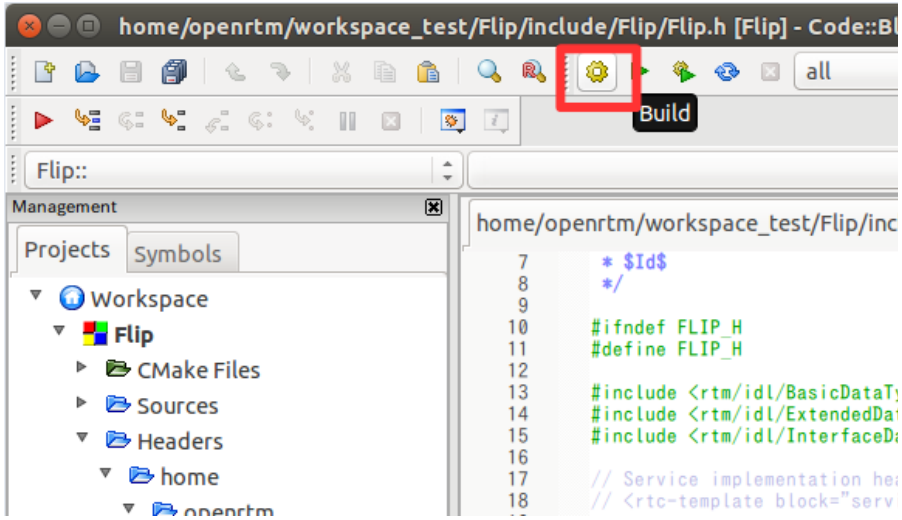


ソースコードのコンパイル

Visual Studio

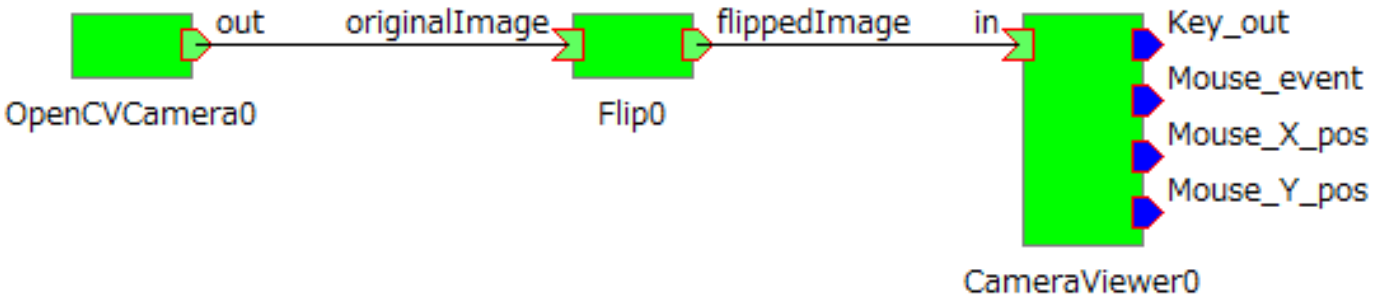


Code::Blocks



Flipコンポーネントの動作確認

- ネーミングサービスを起動する
- RT System Editorを起動する
- CameraViewerコンポーネント、OpenCVCameraコンポーネントを起動する
  - Windows
    - 「OpenRTM-1.1.2」→「C++」→「Components」→「OpenCVExamples」
  - Ubuntu
    - $ /usr/share/openrtm-1.1/components/c++/opencv-rtcs/CameraViewerComp
    - $ /usr/share/openrtm-1.1/components/c++/opencv-rtcs/OpenCVCameraComp
- Flipコンポーネント起動
  - Windows
    - srcフォルダのRelease(もしくはDebug)フォルダ内にFlipComp.exeが生成されているためこれを起動する
  - Ubuntu
    - srcフォルダにFlipCompが生成されているためこれを起動する
- CameraViewerコンポーネント、OpenCVCameraコンポーネント、Flipコンポーネントを接続して「All Activate」を行う



コンフィギュレーションパラメータの操作

- コンフィギュレーションパラメータをRTシステムエディタから操作する