

# WiimoteRTC ドキュメント

# 目次

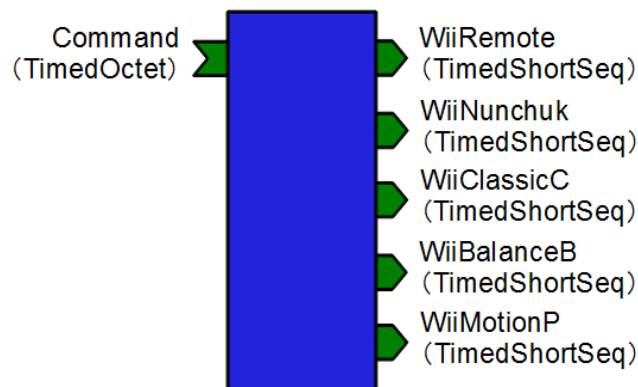
WiimoteRTCドキュメント.....	1
1 コンポーネント概要.....	3
2 仕様.....	3
2.1 動作確認 OS・開発環境.....	3
2.2 依存ライブラリ.....	3
2.3 データポート.....	4
2.4 コンフィギュレーション.....	9
3 WiimoteRTC コンポーネントの利用.....	10
3.1 準備.....	10
3.2 ビルド方法.....	10
3.3 動作確認・開発環境.....	12
4 ライセンス.....	13
5 連絡先.....	13

## 1 コンポーネント概要

本コンポーネントは、任天堂株式会社より発売されている Wii リモコン (以下 WiiRemote) 及びその拡張コントローラの操作及びデータを取得し出力するコンポーネントです。

Command 入力ポートからの入力と、コンフィギュレーションの値を編集することで、WiiRemote リモコンの LED の点灯と振動を操作することができます。

各出力ポートから、WiiRemote リモコンと WiiNunchaku, WiiClassicController, WiiBalanceBoard, WiiMotionPlus の拡張コントローラから取得し、整形したデータを出力します。



※現在、複数の WiiRemote、拡張コントローラを接続することはできません。

## 2 仕様

### 2.1 動作確認 OS・開発環境

- Windows7
- VisualC++2010

### 2.2 依存ライブラリ

- OpenRTM-aist: OpenRTM-aist-1.1.0-RELEASE(C++)
  - wiimote: WiiYourself! - native C++ Wiimote library v1.15  
URL: <http://wiiyourself.gl.tter.org/>
  - WDK7: Windows Driver Kit 7.1.0  
URL: <http://msdn.microsoft.com/ja-jp/library/windows/hardware/hh852365.aspx>
- ※WDK8 は未対応です。

## 2.3 データポート

名前	フローポート	データ型	説明
command	InPort	TimedOuctet	WiiRemote の LED と振動を制御するポートです。上位 4 ビットで LED の点灯パターン、最下位 1 ビットで振動を操作します。
WiiRemote	OutPort	TimedShortSeq	WiiRemote を接続したとき、整形済みデータを送信します。 ※詳細は表1を参照
WiiNunchuk	OutPort	TimedShortSeq	Nunchuk を接続したとき、整形済みデータを送信します。 ※詳細は表2を参照
WiiClassicC	OutPort	TimedShortSeq	ClassicController を接続したとき、整形済みデータを送信します。 ※詳細は表3を参照
WiiBalanceB	OutPort	TimedShortSeq	BalanceBoard を接続したとき、整形済みデータを送信します。 ※詳細は表4を参照
WiiMotionP	OutPort	TimedShortSeq	WiiRemote に MotionPlus を接続したとき、整形済みデータを送信します。 ※詳細は表5を参照

表1 WiiRemote アウトポートの仕様一覧

データ型: TimedShortSeq	
配列番号	詳細
[0]	十字キー・上ボタンの状態(bool 値)
[1]	十字キー・右ボタンの状態(bool 値)
[2]	十字キー・下ボタンの状態(bool 値)
[3]	十字キー・左ボタンの状態(bool 値)
[4]	A ボタンの状態(bool 値)
[5]	B ボタンの状態(bool 値)
[6]	+ ボタンの状態(bool 値)
[7]	Home ボタンの状態(bool 値)
[8]	- ボタンの状態(bool 値)
[9]	1ボタンの状態(bool 値)
[10]	2ボタンの状態(bool 値)
[11]	X 軸方向の加速度 [cm/s <sup>2</sup> ]
[12]	Y 軸方向の加速度 [cm/s <sup>2</sup> ]
[13]	Z 軸方向の加速度 [cm/s <sup>2</sup> ]
[14]	Pitch のデータ [deg]
[15]	Roll のデータ [deg]
[16]	バッテリー残量 [%]
[17]	LED の状態(BYTE 値)

※コンフィギュレーション”SIU\_compati”が”1”の時、[0]～[13]のデータを送信します。

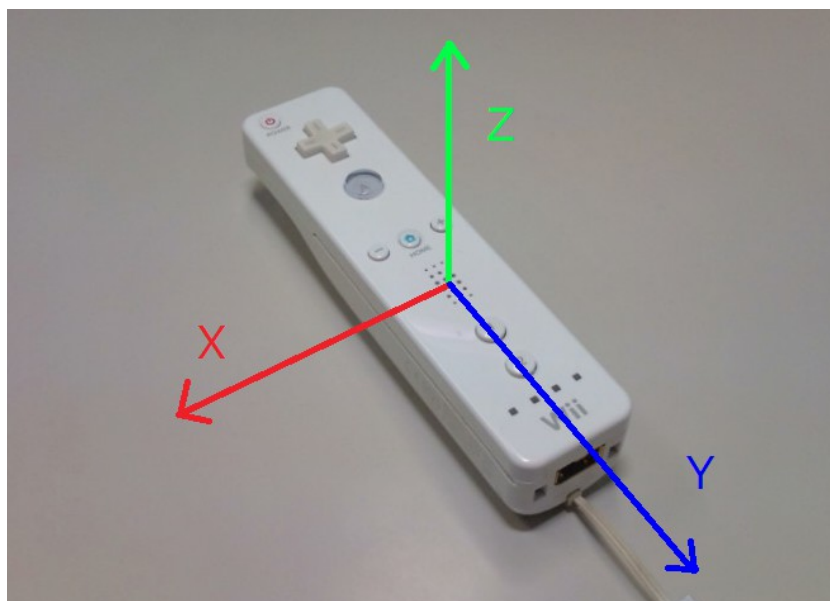


表2 WiiNunchuk アウトポートの仕様一覧

データ型: TimedShortSeq	
配列番号	詳細
[0]～[13]	WiiRemote を参照
[14]	C ボタンの状態(bool 値)
[15]	Z ボタンの状態(bool 値)
[16]	アナログスティックの X 方向入力値 (解放時 0, 最大まで左に傾けると-100, 同右で 100)
[17]	アナログスティックの Y 方向入力値 (解放時 0, 最大まで左に傾けると-100, 同右で 100)
[18]	X 軸方向の加速度 [cm/s <sup>2</sup> ]
[19]	Y 軸方向の加速度 [cm/s <sup>2</sup> ]
[20]	Z 軸方向の加速度 [cm/s <sup>2</sup> ]
[21]	バッテリー残量 [%]
[22]	LED の状態(BYTE 値)

※コンフィギュレーション”SIU\_compati”が”1”の時、[0]～[20]のデータを送信します。

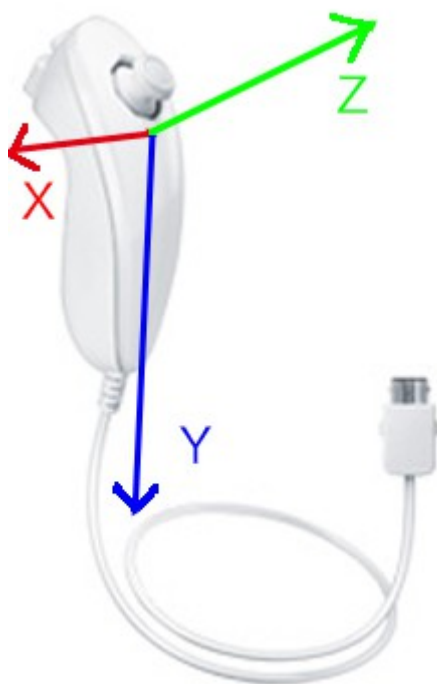
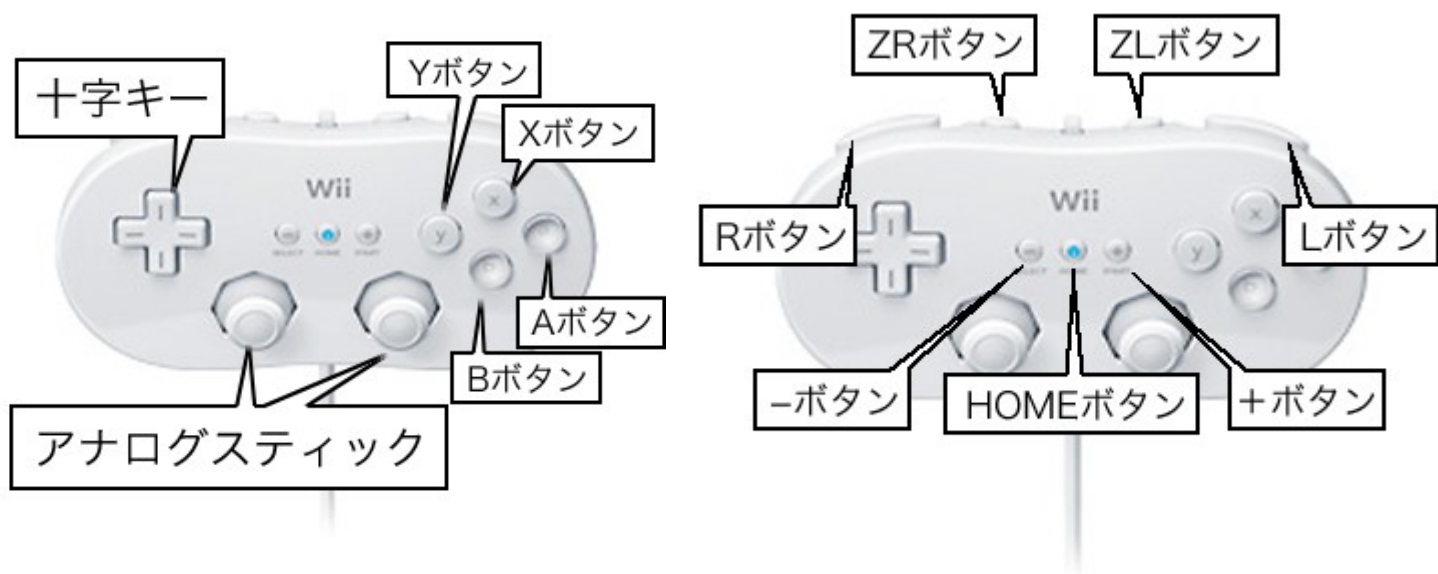


表3 WiiClassicC アウトポートの仕様一覧

データ型: TimedShortSeq	
配列番号	詳細
[0]～[13]	WiiRemote を参照
[14]	A ボタンの状態(bool 値)
[15]	B ボタンの状態(bool 値)
[16]	X ボタンの状態(bool 値)
[17]	Y ボタンの状態(bool 値)
[18]	ーボタンの状態(bool 値)
[19]	Home ボタンの状態(bool 値)
[20]	+ボタンの状態(bool 値)
[21]	十字キー・上ボタンの状態(bool 値)
[22]	十字キー・下ボタンの状態(bool 値)
[23]	十字キー・左ボタンの状態(bool 値)
[24]	十字キー・右ボタンの状態(bool 値)
[25]	L ボタンの状態(bool 値)
[26]	R ボタンの状態(bool 値)
[27]	ZL ボタンの状態(bool 値)
[28]	ZR ボタンの状態(bool 値)
[29]	L ボタンのアナログ入力値 (最小値 0, 最大値 100, PRO の場合は解放で 0, 押下で 100)
[30]	R ボタンのアナログ入力値 (最小値 0, 最大値 100, PRO の場合は解放で 0, 押下で 100)
[31]	左アナログスティックの X 方向入力値 (解放時 0, 最大まで左に傾けると-100, 同右で 100)
[32]	左アナログスティックの Y 方向入力値 (解放時 0, 最大まで下に傾けると-100, 同上で 100)
[33]	右アナログスティックの X 方向入力値(-100～100) (解放時 0, 最大まで左に傾けると-100, 同右で 100)
[34]	右アナログスティックの Y 方向入力値(-100～100) (解放時 0, 最大まで下に傾けると-100, 同上で 100)
[35]	バッテリー残量 [%]
[36]	LED の状態(BYTE 値)

※コンフィギュレーション”SIU\_compati”が”1”の時、[0]～[36]のデータを送信します。

### クラシックコントローラ



### クラシックコントローラ Pro

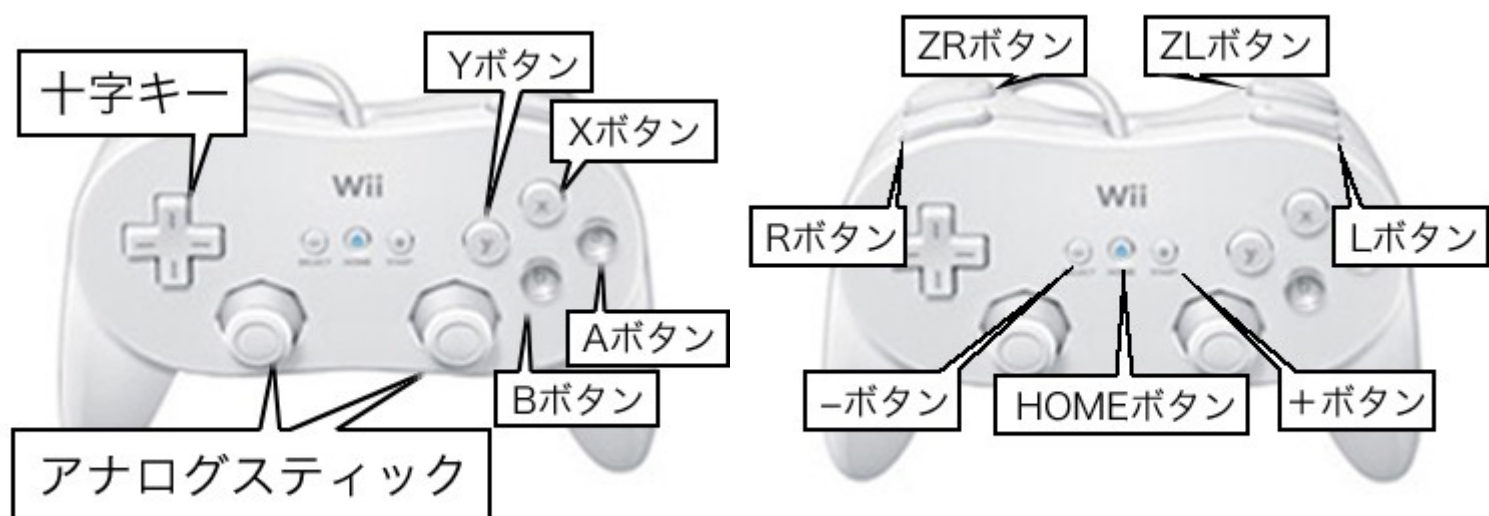




表4 WiiBalanceB アウトポートの仕様一覧

データ型: TimedShortSeq	
配列番号	詳細
[0]	電源ボタンの状態(bool 値)
[1]	X 軸の重心座標
[2]	Y 軸の重心座標
[3]	左上にかかる重心データ
[4]	右上にかかる重心データ
[5]	右下にかかる重心データ
[6]	左下にかかる重心データ
[7]	左上センサの生データ
[8]	右上センサの生データ
[9]	右下センサの生データ
[10]	左下センサの生データ
[11]	バッテリー残量 [%]
[12]	LED の状態(BYTE 値)

※コンフィギュレーション”SIU\_compati”が”1”の時、[0]～[10]のデータを送信します。

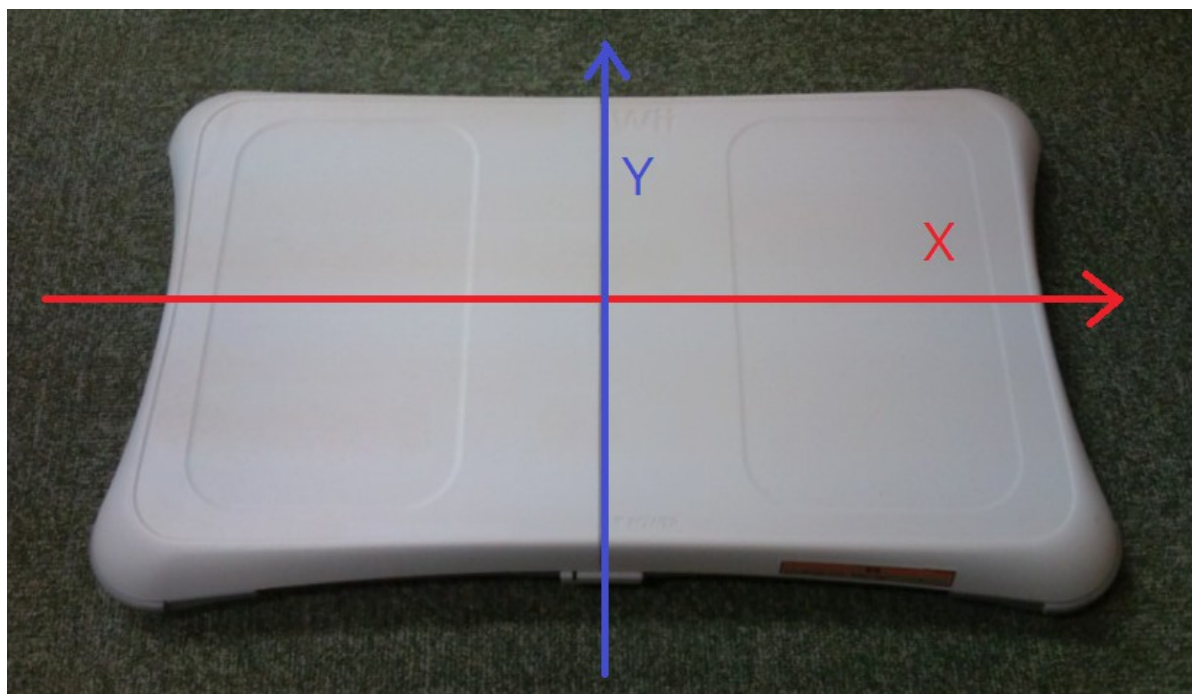
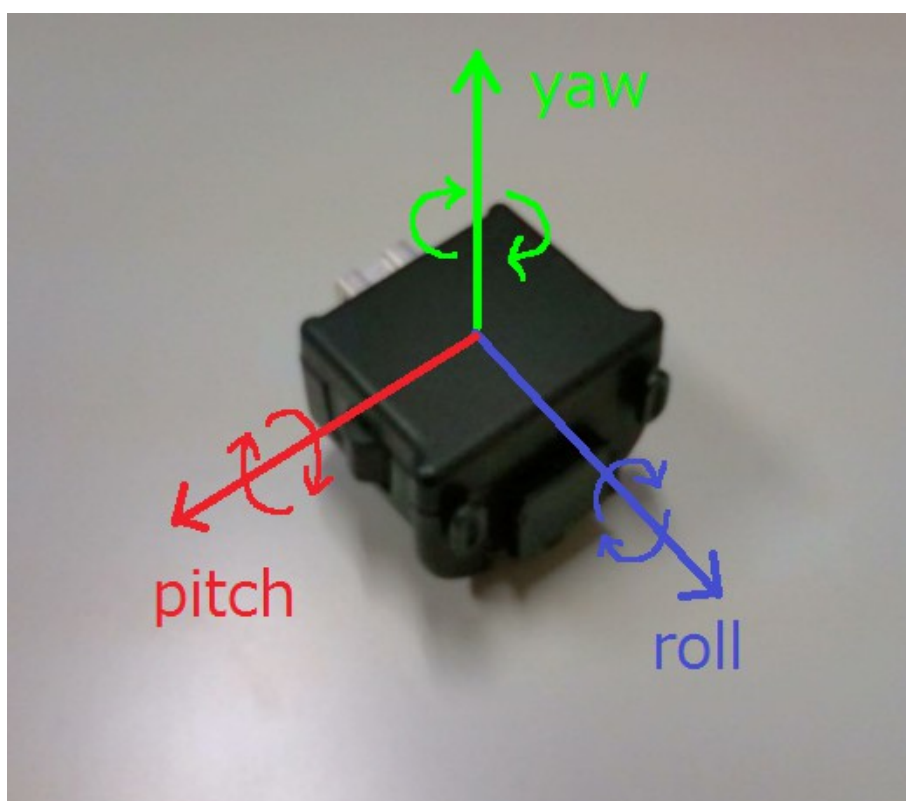


表5 WiiMotionP アウトポートの仕様一覧

データ型: TimedShortSeq	
配列番号	詳細
[0]～[13]	WiiRemote を参照
[14]	MotionPlus の yaw の生データ
[15]	MotionPlus の pitch の生データ
[16]	MotionPlus の roll の生データ
[17]	MotionPlus の yaw を速度に変換した値
[18]	MotionPlus の pitch を速度に変換した値
[19]	MotionPlus の roll を速度に変換した値
[20]	バッテリー残量 [%]
[21]	LED の状態(BYTE 値)

※コンフィギュレーション”SIU\_compati”が”1”の時、[0]～[19]のデータを送信します。



## 2.4 コンフィギュレーション

名前	データ型	デフォルト値	設定範囲	説明
Nunchuk	bool	0	0, 1	拡張コントローラを接続している時に値を”1”にすることで、各アウトポートからデータを出力します。
ClassicC	bool	0	0, 1	
BalanceB	bool	0	0, 1	
MotionP	bool	0	0, 1	
SIU_compati	bool	0	0, 1	芝浦工大殿が開発したWiiRemoteComponentsRTCと同じデータを出力します。
Debug	bool	0	0, 1	コンソール画面に出力データを表示します。
LED	string	無し	0,1,2,3	WiiRemote の4つの LED を左から0～3とし、指定した番号の LED を点灯します。例えば、”0,1”とすると左から1番目と2番目の LED が点灯します。
Rumble	bool	0	0, 1	接続した WiiRemote の振動を操作します。

### 3 WiimoteRTC コンポーネントの利用

ここでは、本コンポーネントのビルド方法について記載します。

#### 3.1 準備

WiimoteRTC コンポーネントに関してコンパイルに必要なライブラリ、ツールは以下の通りです。

##### ● アプリケーション・ツール

- VC++2010
- OpenRTM-aist1.1.0(C++)以降
- cmake
- doxygen
- WindowsDriverKit 7.1.0
- RTSysEditor
- Java Development Kit 6

##### ● ソースコード

WiiYourself! wiimote code by gl.tter

#### 3.2 ビルド方法

(1).最初に環境を整えます。

VC++2010、OpenRTM-aist1.1.0(C++)、cmake、doxygen、WindowsDriverKit 7.1.0 をインストールします。

Java Development Kit 6 をインストールし、RTSysEditor を展開し正常に動作することを確認してください。

WiiYourself!をダウンロードし展開します。

(2).次にコンポーネントに必要な wiimote ライブラリのソースコードを WiimoteRTC ディレクトリに移動します。

展開した WiimoteRTC ディレクトリ直下の wiimote/src ディレクトリに、WiiYourself! の”wiimote.cpp”、”wiimote.h”、”wiimote\_state.h”、”wiimote\_common.h”をコピーします。

以上でソースコードの移動は完了です。

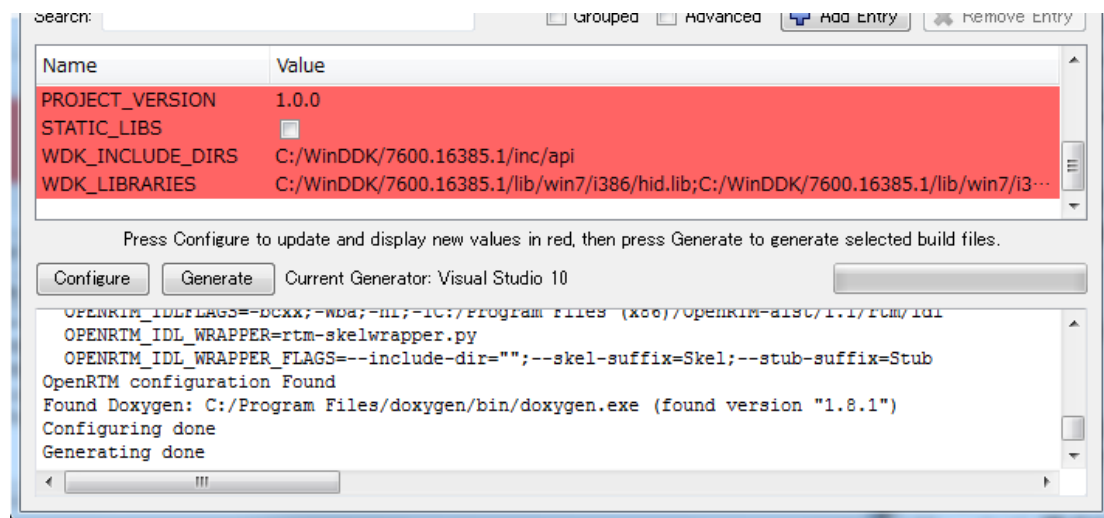
(3).次に、CMake を使って VC++2010 用プロジェクトを作成します。

CMake を起動し、[Where is the source code]に WiimoteRTC ディレクトリ直下にある CMakeLists.txt をドラッグアンドドロップします。または[BrowseSource...]ボタンをクリックして WiimoteRTC ディレクトリを選択します。

[Where to build the binaries]にビルドディレクトリを指定し(場所は任意ですがここでは<ワークディレクトリ>/WiimoteRTC/build のように分かりやすいサブディレクトリを指定します。)、[Configure]ボタンをクリックします。

ウィンドウが表示するので、プロジェクトの種類で[Visual Studio 10]を選択し[Finish]をクリックします。

”configure done”と表示されたら、コンフィギュアは完了です。上のスペースに WDK\_INCLUDE\_DIRS と WDK\_LIBRARIES にパスが入っていることを確認してください。



確認できなかった場合は、WDK 7 のパスを確認し入力します。

WDK\_INCLUDE\_DIRS・・・ <インストール先>WinDDK\7600.16385.1\inc\api

WDK\_LIBRARIES・・・ <インストール先>WinDDK\7600.16385.1\lib\win7\i386

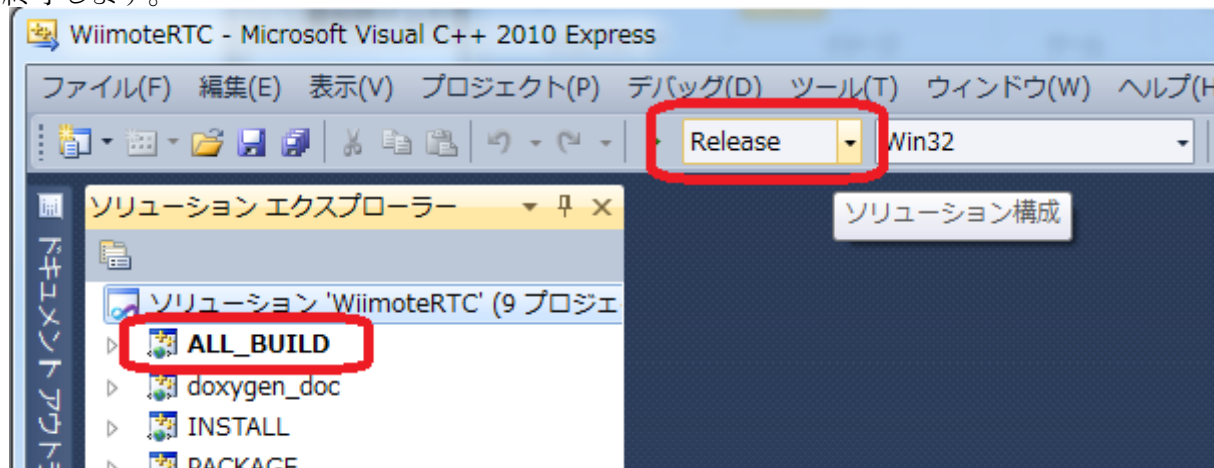
次に、[Generate]ボタンをクリックし、“Generating done”と表示されたらプロジェクト作成は完了です。

CMake は閉じます。

(4).次に、プロジェクトをビルドします。

先ほど指定した build ディレクトリの中にある WiimoteRTC.sln をダブルクリックして Visual C++ 2010 を起動します。

起動後、ソリューション構成を”Debug”から”Release”に変更し、ソリューションエクスプローラーの ALL\_BUILD を右クリックしビルドを選択してビルドします。特に問題がなければ正常にビルドが終了します。



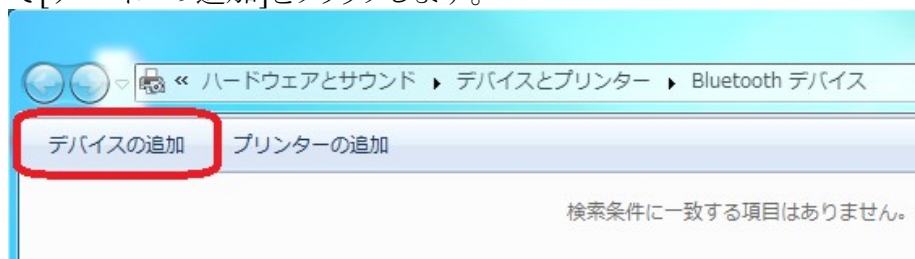
ビルドが正常に終了すると build ディレクトリ以下の/src/Release に WiimoteRTCComp.exe が作成されます。

### 3.3 動作確認・開発環境

完成したコンポーネントを起動し、PC と Bluetooth 接続した WiiRemote の情報を取得したり、LED や振動をコントロールします。

(1).最初に PC と WiiRemote を Bluetooth 接続します。

コントロールパネルの[ハードウェアとサウンド]にある、[Bluetooth デバイス]をクリックし、表示されたウィンドウで[デバイスの追加]をクリックします。



すると[デバイスの追加]ウィンドウが表示され、Bluetooth 接続可能なデバイスが表示されます。

Wiiremote の[1][2]ボタンを同時に押して Bluetooth 接続待ち状態にすると、[デバイスの追加]ウィンドウに次のように表示されます。



※WiiRemote の場合”NintendoRVL-CNT-01”、BalanceBoard の場合”NintendoRVL-WBC-01”と表示されることがあります。

※BalanceBoard を接続する場合、電池カバーの下にある赤い[SYNC]ボタンを押します。

Wiiremote のデバイスを選択し[次へ]をクリックし、次の画面で[ペアリングにコードを使用しない]を選択し接続します。

ウィンドウに”このデバイスは、このコンピュータに正常に追加されました”と表示されたら完了です。ウィンドウは[閉じる]ボタンで閉じてください。

以上で PC と WiiRemote の Bluetooth 接続は完了です。

(2).次に RTSysEditor と Start Naming Service を起動した後、WiimoteRTC コンポーネントを起動します。

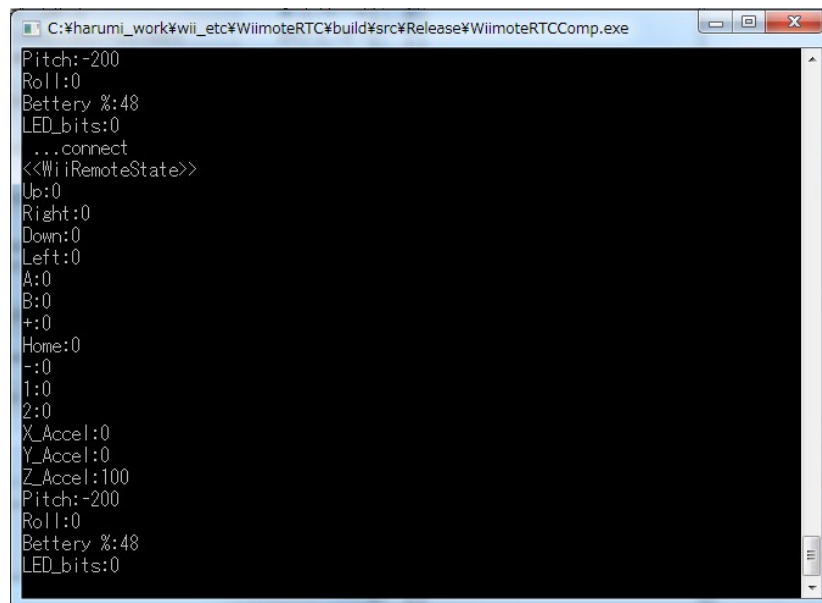
build ディレクトリ以下の/src/Release/に WiimoteRTCComp.exe を起動します。

コンポーネントのコンソール画面が”Connect...OK”と表示したら次に進みます。

”Connect Error...”と表示されたら、WiiRemote との接続に失敗しています。WiiRemote を再度 Bluetooth 接続し、コンポーネントを一旦終了し再起動します。

(3).コンポーネントをシステムダイアグラムに表示しアクティベートします。

下の画像は、コンフィギュレーションの”degug”を”1”に変更し、情報を表示したときのコンソール画面です。

A screenshot of a Windows command prompt window titled "C:\harumi\_work\wii\_etc\WiimoteRTC\build\src\Release\WiimoteRTCComp.exe". The window displays the following text:

```
Pitch:-200
Roll:0
Battery %:48
LED_bits:0
...connect
<<WiiRemoteState>>
Up:0
Right:0
Down:0
Left:0
A:0
B:0
+:0
Home:0
-:0
1:0
2:0
X_Accel:0
Y_Accel:0
Z_Accel:100
Pitch:-200
Roll:0
Battery %:48
LED_bits:0
```

※拡張コントローラや BalanceBoard を使う場合はコンフィギュレーションの値を”1”に変更します。

※WiiRemote の LED や振動をコンフィギュレーションで操作することができます。LED を点灯するときは RTSysEditor のコンフィギュレーションの[編集]ボタンからウィジェットを表示してチェックボックスをクリックします。振動するときは、”Rumble”を”1”に変更します。

以上でコンポーネントの動作確認は完了です。

## 4 ライセンス

WiimoteRTC の著作権は修正 BSD ライセンスです。

マイクロソフト株式会社より提供されている Windows 用 WDK 7 (デバイスドライバ開発キット) と WiiYourself! ソースコード及びライブラリはそれぞれの権利者が定めるライセンスを参照してください。

## 5 連絡先

産業総合技術研究所 知能システム研究部門  
〒305-8568 茨城県つくば市梅園 1-1-1 中央第2  
電話:029-861-5022