Revolution SDK

Wii Balance Board Accessory Programming Manual

Version 1.00

The content of this document is highly confidential and should be handled accordingly.

RVL-06-0345-001-A
Released: March 26, 2008

Table of Contents

1 Overview 6
2 Wii Balance Board Accessory 7
2.1 Balance Sensors 7
2.2 Pairing 7
2.3 Local Regions and Load Values 8
2.4 Sales Format 9
3 Basic Programming 10
3.1 Sample Program: simple_wbc.c 11
3.2 [Japan Only] Sharing Normal Pairing Information 16
3.3 Loading Calibration Data 17
3.4 Get Balance Sensor Output Values 17
3.5 Zero Point Setting 18
3.6 Conversion to Load Values 18
3.7 Corrections 19
4 Measuring Load Accurately 20
4.1 Balance Sensor Properties 20
4.2 Necessity of Temperature Correction and Gravitational Acceleration Correction 20
4.3 Drifting of the Zero Point Setting 20
4.4 Recommended Measurement Procedure 21
5 Troubleshooting Mode 22
6 Battery Level 23
6.1 Battery Level 23
6.2 HOME Menu 23

Code

Code 3-1 simple_wbc.c 11

Tables

Table 2-1 Maximum Weights 9
Table 3-1 Basic APIs 10

Figures

Figure 2-1 The Wii Balance Board accessory Can Be Normal-Paired with Only One Wii Console 8
Figure 2-2 If Normal-Paired with Another Wii Console, Wii Balance Board accessory Settings Are Overwritten 8
Figure 3-1 Library Organization 10
Figure 3-2 Wii Balance Board Connection Flow 15
Figure 3-3 Main Flow 15
Figure 3-4 Correspondence Between Wii Balance Board Sections and Each Foot 17
Figure 5-1 Conceptual Diagram of Troubleshooting Mode (When Only the Lower-Right Section Has Failed) 22

Revision History

Version	Revision Date	Description
1.00	2008/02/26	Initial version.

1 Overview

This document provides the information required to develop software for the Wii Balance Board accessory.

2 Wii Balance Board Accessory

2.1 Balance Sensors

There are four legs attached to the bottom of the Wii Balance Board accessory. A sensor called a “balance sensor,” used to measure load, is attached to each of these legs. When load is applied, a value proportional to that load is returned. The greater the load, the greater this value. Although detailed information about balance sensors is provided in section 3.4, be aware that this value itself has no meaning. It is the amount of change in this value that has meaning. In this context, the “amount of change” refers to the amount of change compared to the state when nothing is on the Wii Balance Board accessory (the state where there is no load applied). A process called “zero point setting” is required to recognize this no-load state. For details on zero point setting, see section 3.5.

The balance of a person (or object) on the Wii Balance Board accessory can be determined by observing the amount of change among these four balance sensors. The load of a person (or object) on the Wii Balance Board accessory is represented by the total value of the amount of change at the four balance sensors.

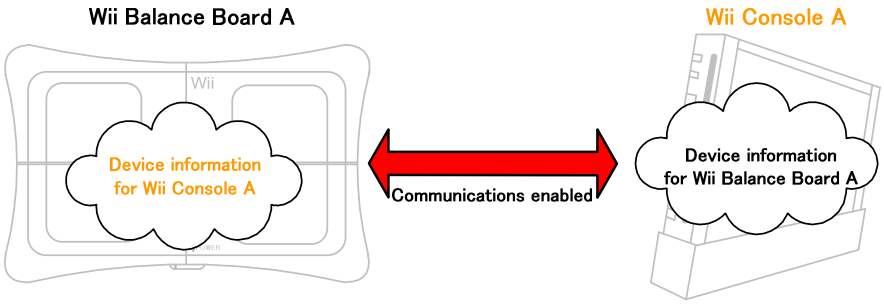
The Wii Balance Board accessory and Wii console communicate via a wireless connection. They cannot communicate via a wired connection. Although wireless communication is carried out at a rate of 200 samples per second, just as with the Wii Remote controller, balance sensor values change at a rate of 60 samples per second.

2.2 Pairing

To use the Wii Balance Board accessory, it is first necessary to perform normal pairing. Pair the Wii Balance Board accessory by simultaneously pressing SYNC on the Wii console and SYNC on the Wii Balance Board accessory. Pairing succeeds when an application that supports the Wii Balance Board accessory is running. Note that pairing on the Wii Menu is impossible because it does not support the Wii Balance Board accessory.

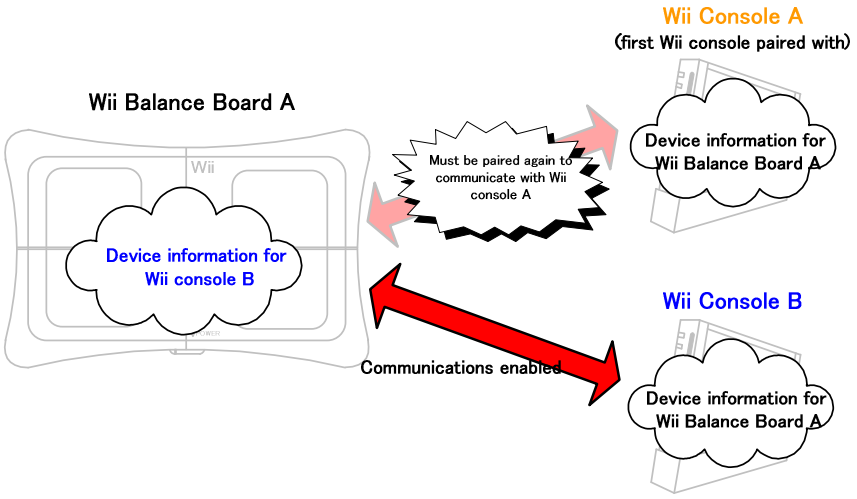
Also note that the Wii Balance Board accessory can be normal-paired with only one Wii console. After normal pairing succeeds, both connected devices carry the other’s device information, and communications between them becomes possible. Similarly, the Wii console can carry the device information for only one Wii Balance Board accessory.

Figure 2-1 The Wii Balance Board accessory Can Be Normal-Paired with Only One Wii Console



If a Wii Balance Board accessory that has already been normal-paired is normal-paired with another Wii console, the settings of the Wii Balance Board accessory are overwritten; to use it with the original Wii Console, normal pairing must be performed again.

Figure 2-2 If Normal-Paired with Another Wii Console, Wii Balance Board accessory Settings Are Overwritten



The Wii Balance Board accessory always connects as Player 4 and is handled in the same manner as a Wii Remote controller. Up to three Wii Remote controllers can therefore be used along with the Wii Balance Board accessory.

2.3 Local Regions and Load Values

There is no local region for the Wii Balance Board accessory itself. The same board is used throughout the world. However, the maximum weight value that can be handled within games does differ depending on the local region. Specifically, maximum load values are as given below.

Table 2-1 Maximum Weights

Japan	America and Europe
Up to 136 kg (300 lbs)	Up to 150 kg (330 lbs)

Because games intended for Japan must observe the Measurement Law, developers must not display a weight on the screen using a numeric value having greater accuracy than that given for the “Weight Display” item listed in the specifications in the *Wii Balance Board Accessory Operations Manual*. For details, see the *Wii Balance Board Accessory Programming Guidelines*.

2.4 Sales Format

As of February 2008, the Wii Balance Board accessory is sold packaged with the Wii Fit video game. There are no plans to sell the Wii Balance Board accessory alone. Therefore, you must consult with Nintendo when considering an application controlled using only the Wii Balance Board accessory.

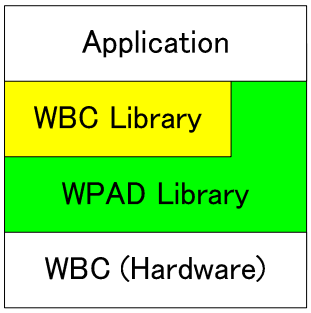
3 Basic Programming

This chapter describes basic programming used to control the Wii Balance Board accessory.

There are two libraries related to the Wii Balance Board accessory: the WPAD library and the WBC library. The WPAD library allows direct access to hardware. The function `WPADRead` is called to get raw output values from the balance sensors.

The WBC library is used to convert values obtained using the `WPADRead` function to load values in kg units. The WBC library is therefore positioned as a higher order library than the WPAD library. Because some functions must be directly controlled from the WPAD library, the library organization associated with the Wii Balance Board accessory is as shown in Figure 3-1 Library Organization.

Figure 3-1 Library Organization



Basic APIs are provided below.

Table 3-1 Basic APIs

Function	Purpose
`WPADRead`	Gets balance sensor output values
`WPADControlBLC`	Issues various control commands to the Wii Balance Board accessory
`WPADRegisterBLCWorkarea`	Configures the work buffers for the library required by the process for sharing registration information for WiiFit for Japan and the Wii Balance Board accessory
`WPADIsRegisteredBLC`	Checks whether a Wii Balance Board accessory is paired with the Wii console
`WBCSetZEROPoint`	Sets the zero point
`WBCRead`	Converts balance sensor output values to load values in kg units
`WBCGetTGCWeight`	Carries out temperature correction and gravitational acceleration correction
WBCSetupCalibration	Loads calibration data*
WBCGetCalibrationStatus	Gets the status of loaded calibration data*

Section 3.1, below, describes simple program code using the basic APIs listed above.

* Data called “calibration data” is stored internally on the Wii Balance Board accessory. This data is used to convert balance sensor output values to weight values in kg units. For details, see section 3.3.

3.1 Sample Program: simple_wbc.c

Code 3-1 simple_wbc.c

```
1: static void ZeroSetStart2( s32 chan, s32 result )
2: {
3:    #pragma unused(chan, result)
4:
5:    // Wait about 200[ms] so that Board's press value rarely becomes unstable
6:    // after updating Balance Wii Board's temperature.
7:    WaitMilliTime(200);
8:
9:    // Set Zero point.
10:    zero[0] = status.press[0]; //Normally, set the average for 2 seconds.
11:    zero[1] = status.press[1]; //Normally, set the average for 2 seconds.
12:    zero[2] = status.press[2]; //Normally, set the average for 2 seconds.
13:    zero[3] = status.press[3]; //Normally, set the average for 2 seconds.
14:    WBCSetZEROPoint( zero,(u32)(sizeof(zero) / sizeof(zero[0])));
15: }
16:
17:
18: static void ZeroSetStart( s32 chan, s32 result )
19: {
20:    #pragma unused(chan, result)
21:
22:    // Check Board's temperature.
23:    WPADRead(WPAD_CHAN3, &status);
24:
25:
26:    if((status.temp == 127) || (status.temp == -128))
27:    {
28:       // Update Board's temperature again if you cannot get correct temperature.
29:       WPADControlBLC(WPAD_CHAN3, WPAD_BLCMD_UPDATE_TEMP, ZeroSetStart2);
30:    }
31:    else
32:    {
33:       // Wait about 200[ms] so that Board's press value rarely becomes unstable
34:       // after updating Balance Wii Board's temperature.
35:       WaitMilliTime(200);
36:
37:       // Set Zero point.
38:       zero[0] = status.press[0]; //Normally, set the average for 2 seconds.
39:       zero[1] = status.press[1]; //Normally, set the average for 2 seconds.
40:       zero[2] = status.press[2]; //Normally, set the average for 2 seconds.
```

```
41:        zero[3] = status.press[3]; //Normally, set the average for 2 seconds.
42:        WBCSetZEROPoint( zero,(u32)(sizeof(zero) / sizeof(zero[0])));
43:    }
44: }
45:
46:
47: int main( void )
48: {
49:    double weight[WPAD_PRESS_UNITS];
50:    u32 type;
51:
52:    DEMOInit( NULL );
53:    DEMOPadInit();
54:
55:    // If Japan, need to call WPADRegisterBLCWorkarea to set a work buffer.
56:    // If not Japan, WPADRegisterBLCWorkarea does nothing.
57:    WPADRegisterBLCWorkarea( workarea );
58:
59:    WPADRegisterAllocator(myAlloc, myFree);
60:    WPADInit();
61:    WPADSetConnectCallback(WPAD_CHAN3, connectCallback);
62:
63:    while (WPAD_STATE_SETUP != WPADGetStatus())
64:    {
65:       ;
66:    }
67:
68:    // You can release the work buffer so initialize of WPAD library has finished.
69:    // This sample program does nothing so it uses static buffer.
70:
71:    while(1)
72:    {
73:       DEMOPadRead();
74:
75:       if (DEMOPadGetButtonDown(0) & PAD_BUTTON_A)
76:       {
77:          // Update Board's temperature the eve of zero point correction.
78:          WPADControlBLC(WPAD_CHAN3, WPAD_BLCMD_UPDATE_TEMP, ZeroSetStart);
79:       }
80:
81:       if (WPADProbe(WPAD_CHAN3, &type) == WPAD_ERR_NO_CONTROLLER)
82:       {
```

```
83:           if (WPADIsRegisteredBLC())
84:           {
85:              OSReport("WBC is not connected.\n");
86:           }
87:           else
88:           {
89:              OSReport("WBC is not registered. Please register WBC with SYNC button.\n");
90:           }
91:       }
92:       else
93:       {
94:           if (type == WPAD_DEV_BALANCE_CHECKER)
95:           {
96:              WPADRead(WPAD_CHAN3, &status);
97:
98:              if(WBCGetCalibrationStatus() == TRUE)
99:              {
100:                 double total;
101:
102:                 WBCRead(&status, weight, (u32)(sizeof(weight) / sizeof(weight[0])));
103:                 total = (double)(weight[0]+weight[1]+weight[2]+weight[3]);
104:
105:                 // Show Total weight. Normally, "total" is the average for 2 seconds.
106:                 OSReport("%3.1f[kg]\n", WBCGetTGCWeight(total, &status));
107:             }
108:             else
109:             {
110:                 OSReport("Calibration data read error. Please reconnect WBC.\n");
111:             }
112:          }
113:          else
114:          {
115:             OSReport("No WBC is attached.\n");
116:          }
117:       }
118:   }
119:}
120:
121:
122:static void connectCallback(s32 chan, s32 reason)
123:{
124:   u32 type;
```

```
125:
126:
127:    if (reason == WPAD_ERR_NONE)
128:    {
129:        // Disconnect 4P if 4P is not Balance Wii board.
130:        // Because 4P is reserved for Balance Wii board.
131:        WPADProbe(chan, &type);
132:        if (chan == WPAD_CHAN3 && type != WPAD_DEV_BALANCE_CHECKER)
133:        {
134:            OSReport("Channel%d is reserved for the balance checker.\n", chan);
135:            WPADDisconnect(chan);
136:        }
137:        else
138:        {
139:            // Read the calibration value for calculating a weight.
140:            if(WPAD_ERR_NONE != WBCSetupCalibration())
141:            {
142:                OSHalt("WBC FATAL ERROR!!\n");
143:            }
144:            OSReport("Channel%d is connected.\n", chan);
145:            WPADSetExtensionCallback(chan, extensionCallback);
146:        }
147:    }
148:    else
149:    {
150:        OSReport("Channel%d is disconnected.\n", chan);
151:    }
152:}
153:
154:
155:static void extensionCallback(s32 chan, s32 result)
156:{
157:    switch(result)
158:    {
159:        case WPAD_DEV_BALANCE_CHECKER:
160:            WPADControlDpd(chan, WPAD_DPD_OFF, NULL);
161:            WPADSetDataFormat(chan, WPAD_FMT_BALANCE_CHECKER);
162:            WPADControlBLC(chan, WPAD_BLCMD_ON, NULL);
163:            OSReport("WBC initialized on channel%d.\n", chan);
164:            break;
165:    }
166:}
```

The processing of this program can be largely divided into two control flows, as shown in Figure 3-2 Wii Balance Board Connection Flow below. Note that the flow shown is intended to show those processes required for the Wii Balance Board accessory in an easy-to-understand manner and does not include all of the control flow in the code.

Figure 3-2 Wii Balance Board Connection Flow illustrates the process executed inside the function registered as the connection callback. This flow occurs when the Wii Balance Board accessory is connected (or disconnected). Figure 3-3 illustrates the main flow.

Figure 3-2 Wii Balance Board Connection Flow

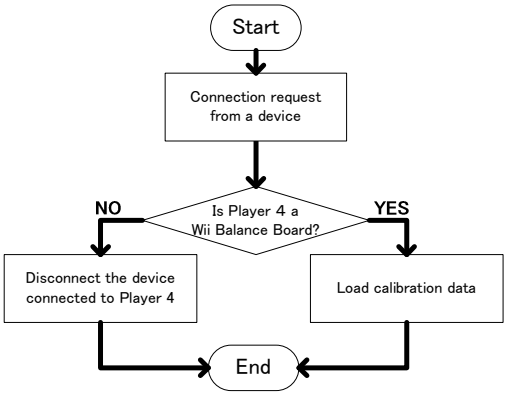
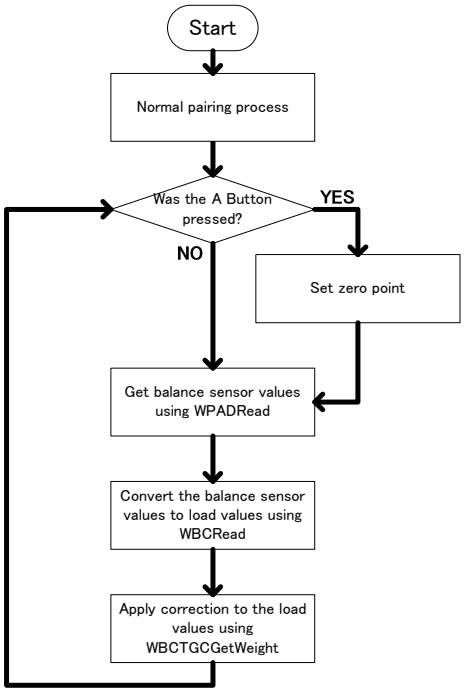


Figure 3-3 Main Flow



A description of each line of the program code is provided below.

Lines 1 to 44: The zero point setting is made here. Although details on setting the zero point are

provided in section 3.5, in simple terms, this process registers the state when nothing is on the Wii Balance Board accessory with the WBC library. The load value at this time is set as the zero point.

Although raw balance sensor output values in the sample code are input for ease of understanding, actually **be sure to obtain balance sensor output values for two seconds (120 samples) and set the average value as the zero point.** (For sample code that uses an average value over two seconds, see `build/demos/wbcdemo/src/handling_weight.c`.)

Lines 57 to 59: This process is used to share pairing information with games that support the Wii Balance Board accessory. This process must be executed before calling the `WPADInit` function (or `KPADInit` function).

Line 96: This line gets balance sensor output values.

Line 102: This line converts balance sensor output values to a load value in kg units.

Line 106: This line applies a correction to the load value. For ease of understanding, the total value of values converted to load values in kg units are input as is in the sample code. **When displaying a load value on the game screen, display a value after obtaining the total value over two seconds (120 samples) and applying correction to that average value.** (For sample code that uses an average value over two seconds, see `build/demos/wbcdemo/src/handling_weight.c`.)

Lines 122 to 152: These lines represent the function registered as the connection callback. This part of code is executed whenever the Wii Balance Board accessory is connected (or disconnected). See Figure 3-2 Wii Balance Board Connection Flow for the process flow. If the device connected to Player 4 is the Wii Balance Board accessory, calibration data is loaded. If the device is not a Wii Balance Board accessory, the device connected to Player 4 is disconnected. **Note that a disconnection process is not executed by the library, even though the Wii Balance Board accessory must be connected to Player 4.**

Lines 155 to 166: These are the functions registered as the extension connection callback. Here, the WPAD data format setting is set to the data format used by the Wii Balance Board accessory.

3.2 [Japan Only] Sharing Normal Pairing Information

Applications intended for Japan must call the `WPADRegisterBLCWorkarea` function before calling the `WPADInit` function or `KPADInit` function. This function executes a process for sharing pairing information of the Wii Balance Board accessory between applications that support the board. The buffer passed to this function must be 32-byte aligned and have a size of at least that given by `WPAD_BLCINT_WORK_LEN`. This buffer can be freed once the `WPADGetStatus` function returns `WPAD_STATE_SETUP`.

Applications intended for regions other than Japan do not need to call the `WPADRegisterBLCWorkarea` function. (However, there is no problem if it is called.)

The `WPADIsRegisteredBLC` function can be used to check if a balance board is already paired.

3.3 Loading Calibration Data

Calibration data specific to the Wii Balance Board accessory is stored internally by the board. This calibration data is used to convert press values to load values in kg units. When the `WBCSetupCalibration` function is called, calibration data is loaded, and a formula for conversion to load values in kg units is created within the WBC library. **Always execute this function after the Wii Balance Board accessory has been linked.**

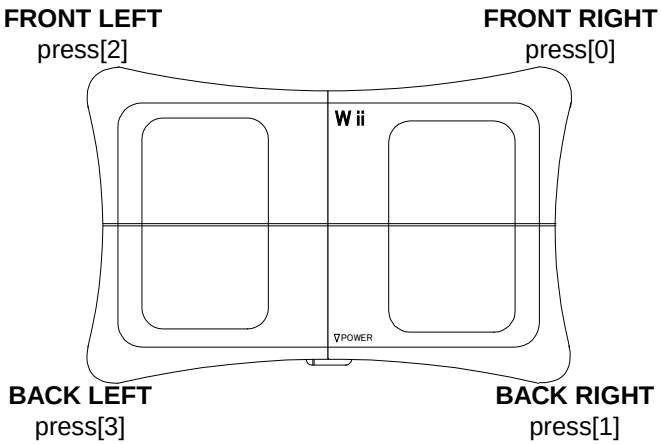
After this function has been executed, each of the APIs, `WBCRead`, `WBCSetZEROPoint`, and `WBCGetTGCWeight`, can be used. (Only the `WBCGetBatteryLevel` function can be used without executing the `WBCSetupCalibration` function beforehand.)

Although the control returns soon after this function executes, the actual process completes after a short while (after approx. 650 ms). Use the `WBCGetCalibrationStatus` to check whether this function has completed normally. If it has not completed normally, execute the `WBCSetupCalibration` function again.

3.4 Get Balance Sensor Output Values

Balance sensor output values are obtained using the `WPADRead` function. A structure specific to WBC (the `WPADBLStatus` structure) is specified to the `WPADRead` function. The `WPADBLStatus` structure stores press member variables. press is a four-element array used to store balance sensor output values. The correspondence between each foot on the actual Wii Balance Board accessory corresponds to the order of press values in the array, as shown in Figure 3-4 Correspondence Between Wii Balance Board Sections and Each Foot.

Figure 3-4 Correspondence Between Wii Balance Board Sections and Each Foot



Observation of press values shows that each value significantly varies from others. For example, the digits and values do not match at all, such as in a case where `press[0]`=1000, `press[1]`=25000, `press[2]`=3700, and `press[3]`=11000. Press values also differ completely depending on the individual Wii Balance Board accessory being used. **Therefore, programs that use the absolute**

value of `press` values are completely meaningless.

The meaning of press values is in the amount of change. In this context, the “amount of change” is from the zero point setting. The zero point setting is described in the next section.

3.5 Zero Point Setting

The zero point setting sets the status when nothing is on the Wii Balance Board accessory for the WBC library. **The zero point setting must be made each time** before measuring load. For details on the importance of making the zero point setting each time, see section 4.1 Balance Sensor Properties.

The zero point setting is made using the function shown below.

```
s32 WBCSetZEROPoint(double press_ave[], u32 size);
```

The two-second average for each press value obtained by the `WPADRead` function is set in the argument `press_ave`. Be sure to set this argument in the same order as the press values. (For example, assign the average of `status.press[1]` to `press_ave[1]`. Set size to the number of elements in the array `press_ave`. (Normally, allocate for four values.)

Before sampling the two-second average for each press value, update temperature information for the Wii Balance Board accessory using the temperature update command (`WPADControlBLC` function). After updating the temperature, use the `WPADread` function to check whether the temperature has been updated correctly.

A value of `-128` or `127` is very occasionally returned. If this happens, execute the temperature update command (`WPADControlBLC` function) again.

Wii Balance Board press values very occasionally fluctuate for a short while after the update command is sent. Therefore, be sure to obtain press values after waiting a short while (approx. 200 ms) after issuing the temperature update command.

To determine whether anything is on the Wii Balance Board accessory when setting the zero point, check the return value of the `WBCRead` function.

3.6 Conversion to Load Values

The following API (`WBCRead`) is used to convert to load values in kg units.

```
s32 WBCRead(WPADBLStatus *status, double weight[], u32 size);
```

The structure `WPADBLStatus` obtained by the `WPADRead` function is set as an argument. The value converted to a load value in kg units is stored in `weight` in the same order as press values. Specify the number of elements in the `weight` array in size. (Normally, allocate for four values.)

3.7 Corrections

The following API (`WBCGetTGCWeight`) is used to correct both temperature and gravitational acceleration.

```
double WBCGetTGCWeight(double total_weight_ave, WPADBLStatus *status);
```

The two-second average for the total of the four load values obtained using the `WBCRead` function is set in the argument `total_weight_ave`. **Always apply correction if the load value is actually to be used.** For the importance of performing temperature correction and gravitational acceleration correction, see section 4.2 Necessity of Temperature Correction and Gravitational Acceleration Correction.

4 Measuring Load Accurately

4.1 Balance Sensor Properties

Balance sensors exhibit “hysteresis” characteristics wherein, once load is applied, measured values do not completely return to their original state even if that load is reduced. For example, take a case where the zero point is set, a person (or object) gets on the Wii Balance Board accessory and then gets off. The status of the Wii Balance Board accessory at this time is not exactly the same as when the zero point was originally set.

The status when no load is applied to balance sensors constantly varies each time a person (or object) gets on the board. For this reason, **it is necessary to set the zero point every time before making accurate measurements.**

4.2 Necessity of Temperature Correction and Gravitational Acceleration Correction

Balance sensors are metallic (aluminum). Metals can be deformed more easily, or less easily, depending on the temperature. Specifically, metals are more easily deformed at higher temperatures and less easily at lower ones. When the metal deforms more easily, it causes load values to be slightly larger, and when the metal deforms less easily, it causes load values to be slightly smaller.

Weight varies in proportion to gravity. It is therefore necessary to consider gravitational acceleration when dealing with load values. Gravitational acceleration varies slightly due to latitude. Specifically, gravitational acceleration is smaller at zero latitude (on the equator) because centrifugal force due to the Earth’s own rotation is greatest there. As a result, load values at the equator are smaller, while weight values at the North and South Poles are greater.

The `WBCGetTGCWeight` function performs both temperature correction and gravitational acceleration correction. Always use the `WBCGetTGCWeight` function to correct load values when performing accurate measurements.

4.3 Drifting of the Zero Point Setting

Output from balance sensors varies due to factors such as the time elapsed since the board was turned on. The zero point can therefore drift if it is not updated even without a person (or object) getting on and off. **For accurate measurements, it is therefore necessary to measure load soon after setting the zero point** in cases such as displaying a person’s body weight.

4.4 Recommended Measurement Procedure

The ideal process flow for accurate measurements is as follows.

1. Prompt the user to get off the Wii Balance Board accessory.
2. Set the zero point with no load on the board.
3. Have the user get on the board quickly and measure the load.
4. Correct the measured load value using the correction function.

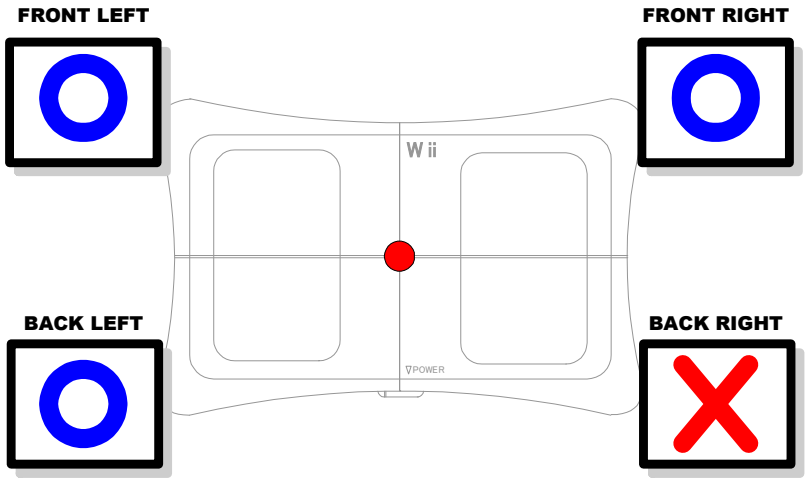
Always use this procedure for every measurement.

5 Troubleshooting Mode

Games that support the Wii Balance Board accessory must implement a Troubleshooting Mode. This is necessary so that the user can confirm whether the Wii Balance Board accessory is operating normally.

In Troubleshooting Mode, check whether it is possible to obtain measurement values from each of the four sensors of the Wii Balance Board accessory. The board fails the check if a measurement value cannot be obtained for even one sensor. See Figure 5-1 Conceptual Diagram of Troubleshooting Mode (When Only the Lower-Right Section Has Failed) for the screen used during Troubleshooting Mode.

Figure 5-1 Conceptual Diagram of Troubleshooting Mode (When Only the Lower-Right Section Has Failed)



The recommended procedure is as follows.

1. First, always check the battery level using the `WBCGetBatteryLevel` function. If zero is returned, prompt the user to replace the batteries.
2. Have the user get off the Wii Balance Board accessory and set the zero point.
3. Once the zero point is set, have the user get on the Wii Balance Board accessory within 10 seconds and have them apply load evenly.
4. Measure the load (two-second average). At this time, check for change in the load values for each foot. Operations are normal if the change is greater than or equal to 2 kg; otherwise, operations are abnormal. Display the result of the troubleshooting (for example, OK or NG) to inform the user whether the Wii Balance Board accessory is broken.

As the method for checking whether Troubleshooting Mode is implemented correctly, get on one of the four corners of the Wii Balance Board accessory and check whether the board fails the check.

6 Battery Level

6.1 Battery Level

The balance sensors located at the four corners of the Wii Balance Board accessory stop operating when the battery level falls below a certain threshold. However, since the wireless module continues operating, the Wii Balance Board accessory continues to send data. If the battery level falls below the threshold value, all press values will go to zero, and, a value of zero will be sent continuously as a result.

The `WBCGetBatteryLevel` function returns zero in this state. Applications should use this function to check the battery level and prompt the user to replace batteries if zero is returned.

6.2 HOME Menu

Use the library especially intended for the Wii Balance Board accessory for the HOME Menu library. This is because code for processing the battery level differs from that used with the Wii Remote controller.