Revolution SDK
Wii Balance Board Accessory Programming Manual

Version 1.2

RVL-06-0345-001-D
Released: November 3, 2008

Table of Contents

Code

Tables

Figures

Revision History

Version	Revision Date	Description
1.2	2008/10/28	Corrected Figure 5-1.
	2008/09/09	Added to Chapter 5 that the trouble shooting mode was implemented as a separate mode. Added Section 6.2. Added information related to retrying the loading of calibration data to Section 3.3. Changed the notation for the `WBCGetTGCWeight` and `WBCSetupCalibration` functions in Section 3.1. Changed the notation for the `WBCGetTGCWeight` function in Section 3.7.
1.1	2008/04/02	Section 2.2 Pairing: Added a precaution about games that support four players. Section 2.3 Regions and Load Values: • Explained that in Japan, if the load exceeds 136 kg, usage is permitted up to 150 kg as long as the load is not displayed. • Added a precaution about minimum loads. Chapter 5 Troubleshooting Mode • Noted that all four corners have to be checked. • Deleted “Troubleshooting Mode Screen." • Added Figure 5-1 Troubleshooting Mode Flow and changed the troubleshooting procedures to match. Deleted section 6.2.
1.0	2008/02/26	Initial version.

1 Overview

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

2 Wii Balance Board Accessory

2.1 Balance Sensors

Four legs are attached to the bottom of the Wii Balance Board accessory. A sensor, called the *balance sensor*, used to measure load, is attached to each of these legs. When a 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 Get Balance Sensor Output Values, 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 *no-load state*). A method called *zero point setting* (see section 3.5 Zero Point Setting) is required to recognize this no-load state.

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

The Wii Balance Board accessory and Wii console can communicate only via a wireless 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.

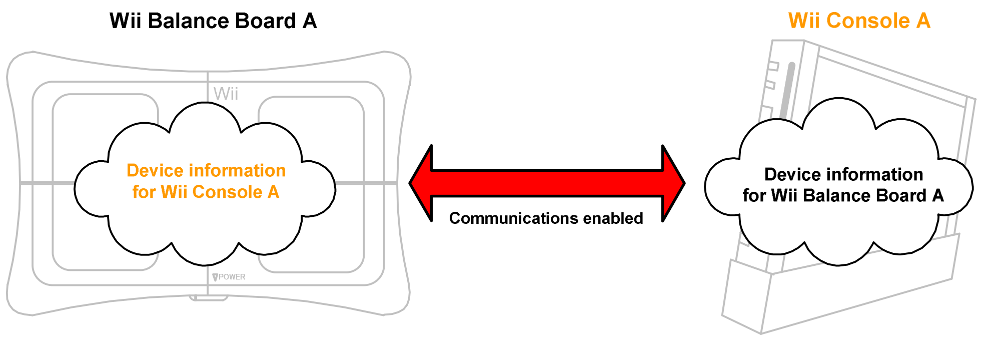
2.2 Pairing

To use the Wii Balance Board accessory, it is first necessary to perform normal pairing with the Wii console by simultaneously pressing the SYNC buttons on both. Pairing succeeds when an application that supports the Wii Balance Board accessory is running. Note that pairing from 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.

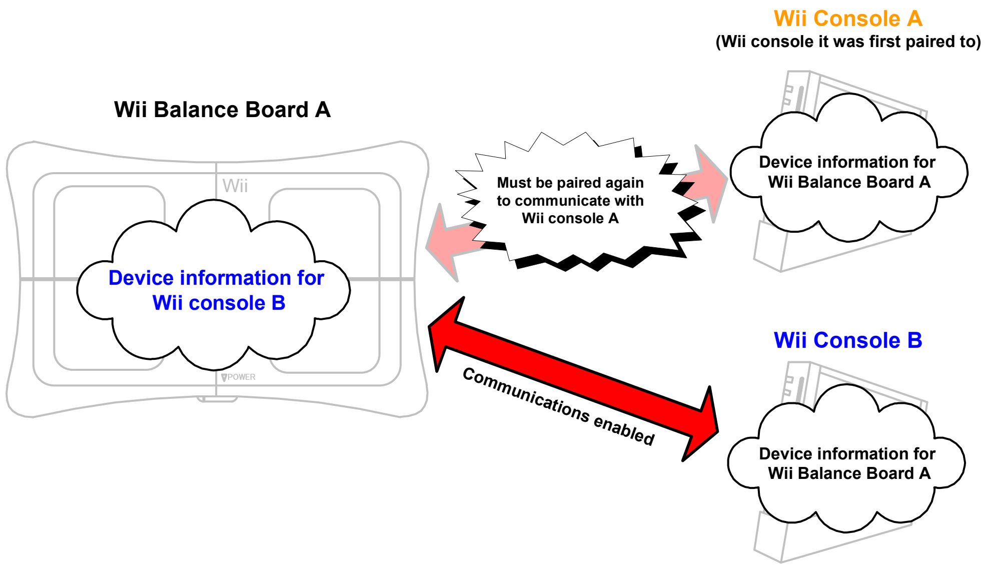
After normal pairing succeeds, both connected devices carry the other’s device information, and communications between them becomes possible. The Wii Balance Board accessory can carry the information for only one Wii console. Similarly, the Wii console can carry the device information for only one Wii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

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


Wii Balance Board A
Wii Console A
Device information for Wii Console A
Communications enabled
Device information for Wii Balance Board A

If a Wii Balance Board accessory that has already been normal-paired is normal-paired with another Wii console, the Wii Balance Board accessory's settings will be overwritten (see Figure 2-2). To use it with the original Wii console, normal-pairing must be repeated.

Figure 2-2 Normal-Pairing with Another Wii Console Overwrites the Wii Balance Board Accessory's Settings


Wii Console A
(Wii console it was first paired to)
Wii Balance Board A
Must be paired again to communicate with Wii console A
Device information for Wii Balance Board A
Device information for Wii console B
Wii Console B
Communications enabled
Device information for Wii Balance Board A

The Wii Balance Board accessory always connects as Player 4 and is handled in the same manner as the Wii Remote controller. Therefore, up to three Wii Remote controllers can be used along with the Wii Balance Board accessory. This means, for example, that if four Wii Remote controllers are being used in a game that supports up to four players, and part way through the game a Wii Balance Board accessory is used, Player 4's Wii Remote controller will become unusable, even though it was used normally up to that point. Because this may confuse the user, be sure to implement some measure, such as displaying a cautionary message, that is appropriate for each game.

2.3 Regions and Load Values

The Wii Balance Board accessory is not market-specific. The same accessory is used throughout the world. However, the maximum load value that can be handled in a game does differ depending on the local market. Maximum load values are shown in Table 2-1.

Table 2-1 Maximum Load

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

Because games intended for sale in Japan must observe the Measurement Law, the numeric load values displayed on the screen cannot have an accuracy greater than that detailed in the “Weight Display” specification listed in the *Wii Balance Board Accessory Operations Manual*. However, even if a user exceeds this weight limit, if the weight is not displayed on the screen, the accessory can be used as long as the load limit (150 kg) is not exceeded. For more details, see section 2.2 “Load Restrictions [Required]” in the *Wii Balance Board Programming Guidelines*. Set the minimum load to 0 kilograms; do not create a recommended minimum load value that is greater than zero.

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.

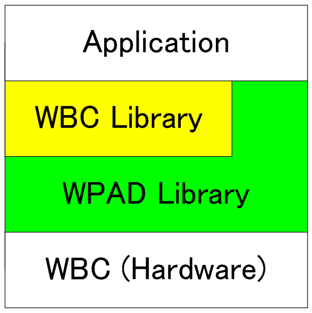
3 Basic Programming

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

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

The WBC library is used to convert values obtained by `WPADRead` into load values, in kilogram units. Therefore, it is 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.

Figure 3-1 Library Organization



Basic library functions are provided below.

Table 3-1 Basic Functions

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 Wii Fit for Japan and the Wii Balance Board accessory.
`WPADIsRegisteredBLC`	Checks whether the 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 kilograms.
`WBCGetTGCWeight`	Corrects temperature and gravitational acceleration.
`WBCSetupCalibration`	Loads calibration data*.
`WBCGetCalibrationStatus`	Gets the status of loaded calibration data.

The following section describes simple program code using the basic functions listed above.

* Data called *calibration data* is stored internally on the Wii Balance Board accessory. This data is used to convert balance sensor output to weight values in kilogram units and is unique to each individual Wii Balance Board accessory. For details, see section 3.3 Loading Calibration Data.

3.1 Sample Program: simple_wbc.c

Code 3-1 `simple_wbc.c`

```c
static void ZeroSetStart2( s32 chan, s32 result )
{
    #pragma unused(chan, result)

    // Wait about 200[ms] so that Board's press value rarely becomes unstable
    // after updating Balance Wii Board's temperature.
    WaitMilliTime(200);

    // Set Zero point.
    zero[0] = status.press[0]; //Normally, set the average for 2 seconds.
    zero[1] = status.press[1]; //Normally, set the average for 2 seconds.
    zero[2] = status.press[2]; //Normally, set the average for 2 seconds.
    zero[3] = status.press[3]; //Normally, set the average for 2 seconds.
    WBCSetZEROPoint( zero,(u32)(sizeof(zero) / sizeof(zero[0])));
}


static void ZeroSetStart( s32 chan, s32 result )
{
    #pragma unused(chan, result)

    // Check Board's temperature.
    WPADRead(WPAD_CHAN3, &status);


    if((status.temp == 127) || (status.temp == -128))
    {
        // Update Board's temperature again if you cannot get correct temperature.
        WPADControlBLC(WPAD_CHAN3, WPAD_BLCMD_UPDATE_TEMP, ZeroSetStart2);
    }
    else
    {
        // Wait about 200[ms] so that Board's press value rarely becomes unstable
        // after updating Balance Wii Board's temperature.
        WaitMilliTime(200);

        // Set Zero point.
        zero[0] = status.press[0]; //Normally, set the average for 2 seconds.
        zero[1] = status.press[1]; //Normally, set the average for 2 seconds.
        zero[2] = status.press[2]; //Normally, set the average for 2 seconds.
        zero[3] = status.press[3]; //Normally, set the average for 2 seconds.
        WBCSetZEROPoint( zero,(u32)(sizeof(zero) / sizeof(zero[0])));
    }
}


int main( void )
{
    double weight[WPAD_PRESS_UNITS];
    u32 type;

    DEMOInit( NULL );
    DEMOPadInit();

    // If Japan, need to call WPADRegisterBLCWorkarea to set a work buffer.
    // If not Japan, WPADRegisterBLCWorkarea does nothing.
    WPADRegisterBLCWorkarea( workarea );

    WPADRegisterAllocator(myAlloc, myFree);
    WPADInit();
    WPADSetConnectCallback(WPAD_CHAN3, connectCallback);

    while (WPAD_STATE_SETUP != WPADGetStatus())
    {
        ;
    }

    // You can release the work buffer so initialize of WPAD library has finished.
    // This sample program does nothing so it uses static buffer.

    while(1)
    {
        DEMOPadRead();

        if (DEMOPadGetButtonDown(0) & PAD_BUTTON_A)
        {
            // Update Board's temperature the eve of zero point correction.
            WPADControlBLC(WPAD_CHAN3, WPAD_BLCMD_UPDATE_TEMP, ZeroSetStart);
        }

        if (WPADProbe(WPAD_CHAN3, &type) == WPAD_ERR_NO_CONTROLLER)
        {
            if (WPADIsRegisteredBLC())
            {
                OSReport("WBC is not connected.\n");
            }
            else
            {
                OSReport("WBC is not registered. Please register WBC with SYNC button.\n");
            }
        }
        else
        {
            if (type == WPAD_DEV_BALANCE_CHECKER)
            {
                WPADRead(WPAD_CHAN3, &status);

                if(WBCGetCalibrationStatus() == TRUE)
                {
                    double total;
                    double tgc_weight;

                    WBCRead(&status, weight, (u32)(sizeof(weight) / sizeof(weight[0])));
                    total = (double)(weight[0]+weight[1]+weight[2]+weight[3]);

                    // Show Total weight. Normally, "total" is the average for 2 seconds.
                    if(WBCGetTGCWeight(total, &status, &tgc_weight)== WBC_ERR_NONE)
                    {
                        OSReport("%3.1f[kg]\n",tgc_weight);
                    }
                }
                else
                {
                    OSReport("Calibration data read error. Please reconnect WBC.\n");
                }
            }
            else
            {
                OSReport("No WBC is attached.\n");
            }
        }
    }
}
```

```

static void connectCallback(s32 chan, s32 reason)
{
    u32 type;

    if (reason == WPAD_ERR_NONE)
    {
        // Disconnect 4P if 4P is not Wii Balance Board.
        // Because 4P is reserved for Wii Balance Board.
        WPADProbe(chan, &type);
        if (chan == WPAD_CHAN3 && type != WPAD_DEV_BALANCE_CHECKER)
        {
            OSReport("Channel%d is reserved for the balance checker.\n", chan);
            WPADDisconnect(chan);
        }
        else
        {
            // Read the calibration value for calculating a weight.
            if(WBCSetupCalibration())
            {
                OSHalt("WBC FATAL ERROR!!\n");
            }
            OSReport("Channel%d is connected.\n", chan);
            WPADSetExtensionCallback(chan, extensionCallback);
        }
    }
    else
    {
        OSReport("Channel%d is disconnected.\n", chan);
    }
}


static void extensionCallback(s32 chan, s32 result)
{
    switch(result)
    {
        case WPAD_DEV_BALANCE_CHECKER:
            WPADControlDpd(chan, WPAD_DPD_OFF, NULL);
            WPADSetDataFormat(chan, WPAD_FMT_BALANCE_CHECKER);
            WPADControlBLC(chan, WPAD_BLCMD_ON, NULL);
            OSReport("WBC initialized on channel%d.\n", chan);
            break;
    }
}
```

The processing of this program can be largely divided into two control flows, as shown in Figure 3-2 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 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 Accessory Connection Flow

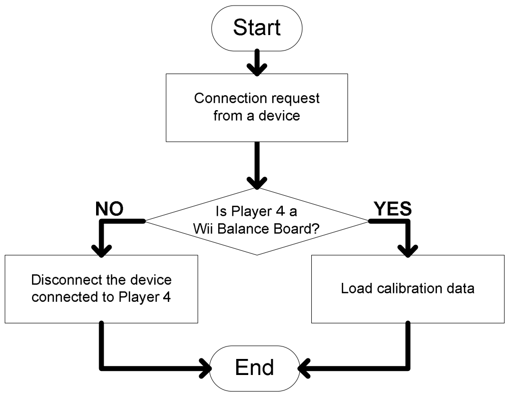
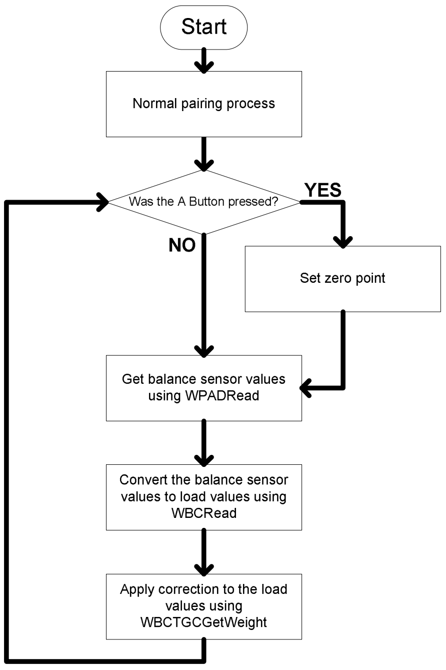

Start
Connection request from a device
NO
Is Player 4 a Wii Balance Board?
YES
Disconnect the device connected to Player 4
Load calibration data
End

Figure 3-3 Main Flow


Start
Normal pairing process
Was the A Button pressed?
YES
NO
Set zero point
Get balance sensor values using WPADRead
Convert the balance sensor values to load values using WBCRead
Apply correction to the load values using WBCTGCGetWeight

Parts of the program code are detailed 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, **be sure to obtain balance sensor output values for two seconds (120 samples) and set the average of those values 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 103: This line converts balance sensor output values to load values in kg units.

Line 107: 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 is input unchanged in the sample code. **When displaying a load value on the game screen, display a value after obtaining the total values over two seconds (120 samples) and applying correction to the average of those totals.** (For sample code that uses an average value over two seconds, see `build/demos/wbcdemo/src/handling_weight.c`.)

Lines 1262 to 155: 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 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 158 to 169: 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 is used to verify that a balance board is already paired.

3.3 Loading Calibration Data

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

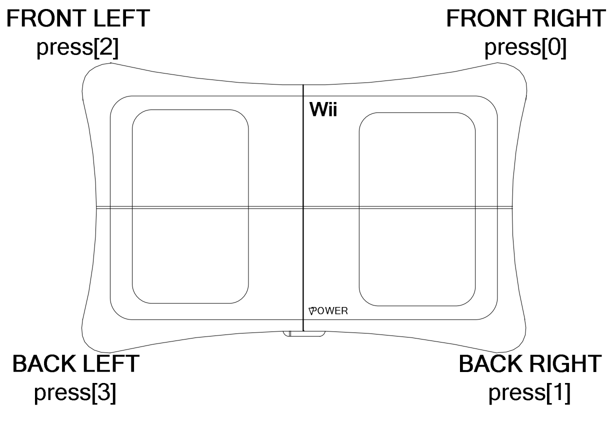
After this function has been executed, the `WBCRead`, `WBCSetZEROPoint`, and `WBCGetTGCWeight` functions can all be used. (Only the `WBCGetBatteryLevel` function can be used without executing the `WBCSetupCalibration` function beforehand.)

Although the control returns soon after the `WBCSetupCalibration` function executes, the actual process completes after a short while (approximately 650 ms). Use the `WBCGetCalibrationStatus` function to check whether this function has completed normally; if it has not completed normally after more than one second has passed, execute the `WBCSetupCalibration` function again. If it fails three consecutive times, decide that a malfunction has occurred and display to the screen a message such as "The Wii Balance Board has malfunctioned. Please read the Wii Balance Board user's manual for further information."

3.4 Get Balance Sensor Output Values

Balance sensor output values are obtained using the `WPADRead` function. `WPADBLStatus` is a WBC-specific structure; it is specified to `WPADRead` and it stores the `press` member variables. The `press` array is a four-element array used to store balance sensor output values. Figure 3-4 shows the correspondence between each foot of the Wii Balance Board accessory and the order of `press` values in the array.

Figure 3-4 Correspondence Between Wii Balance Board Accessory's press Values and Each Foot



Observation of `press` values shows that each value varies significantly from others. For example, the number of digits and their 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

In the WBC library the zero point setting sets the state where nothing is on the Wii Balance Board accessory. The zero point setting must be set every time before you measure a load. For details on the importance of making the zero point setting every time, see section 4.1 Balance Sensor Properties.

The zero point setting is determined with the following function:

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

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

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

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

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

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

3.6 Conversion to Load Values

The `WBCRead` function converts raw values into load values in kilogram units.

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

The `WPADBLStatus` structure, which is obtained by `WPADRead`, is set as an argument. The converted load values are stored in `weight` in the same array order as the `press` values. Set `size` to the number of elements in the `weight` array (normally allocated for four values).

3.7 Corrections

The `WBCGetTGCWeight` function corrects for both temperature and gravitational acceleration.

```
s32 WBCGetTGCWeight(double total_weight_ave, double *tgc_weight, WPADBLStatus *status);
```

The two-second average for the totals of the four load values obtained using the `WBCRead` function is set in the `total_weight_ave` argument. The value after correction is stored in the `tgc_weight` argument. **Always apply correction if you are going to use the load value.** For more information on the importance of correcting for temperature and gravitational acceleration, 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, assume the zero point is set, and then a person (or object) steps on the Wii Balance Board accessory and then steps off. The status of the Wii Balance Board accessory after the load is removed 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) steps on the board. Therefore, to ensure accuracy, **you must set the zero point each time before taking 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 load values at the North and South Poles are greater.

The `WBCGetTGCWeight` function performs both temperature correction and gravitational acceleration correction. To ensure accurate measurements, always use the `WBCGetTGCWeight` function to correct load values.

4.3 Drifting of the Zero Point Setting

Output from the balance sensors varies due to factors such as the time elapsed since the Wii Balance Board accessory was turned on. The zero point can drift if it is not updated, even if no load was placed on or removed from the accessory. For accurate measurements, such as when measuring and displaying a person’s weight, the load must be measured soon after the zero point is set.

4.4 Recommended Measurement Procedure

To obtain accurate measurements, use the following procedure.

<u>Use this procedure for every measurement.</u>

1. Prompt the user to step off the Wii Balance Board accessory.
2. With no load on the accessory, set the zero point.
3. Immediately prompt the user to step onto the accessory and measure the load.
4. Correct the measured load value using the correction function.

5 Troubleshooting Mode

Games that support the Wii Balance Board accessory must implement a troubleshooting mode, which allows the user to verify that the Wii Balance Board is working normally. Because the purpose of this mode is to allow the user to troubleshoot voluntarily, implement troubleshooting mode as a separate mode from the game.

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.

Troubleshooting mode procedure

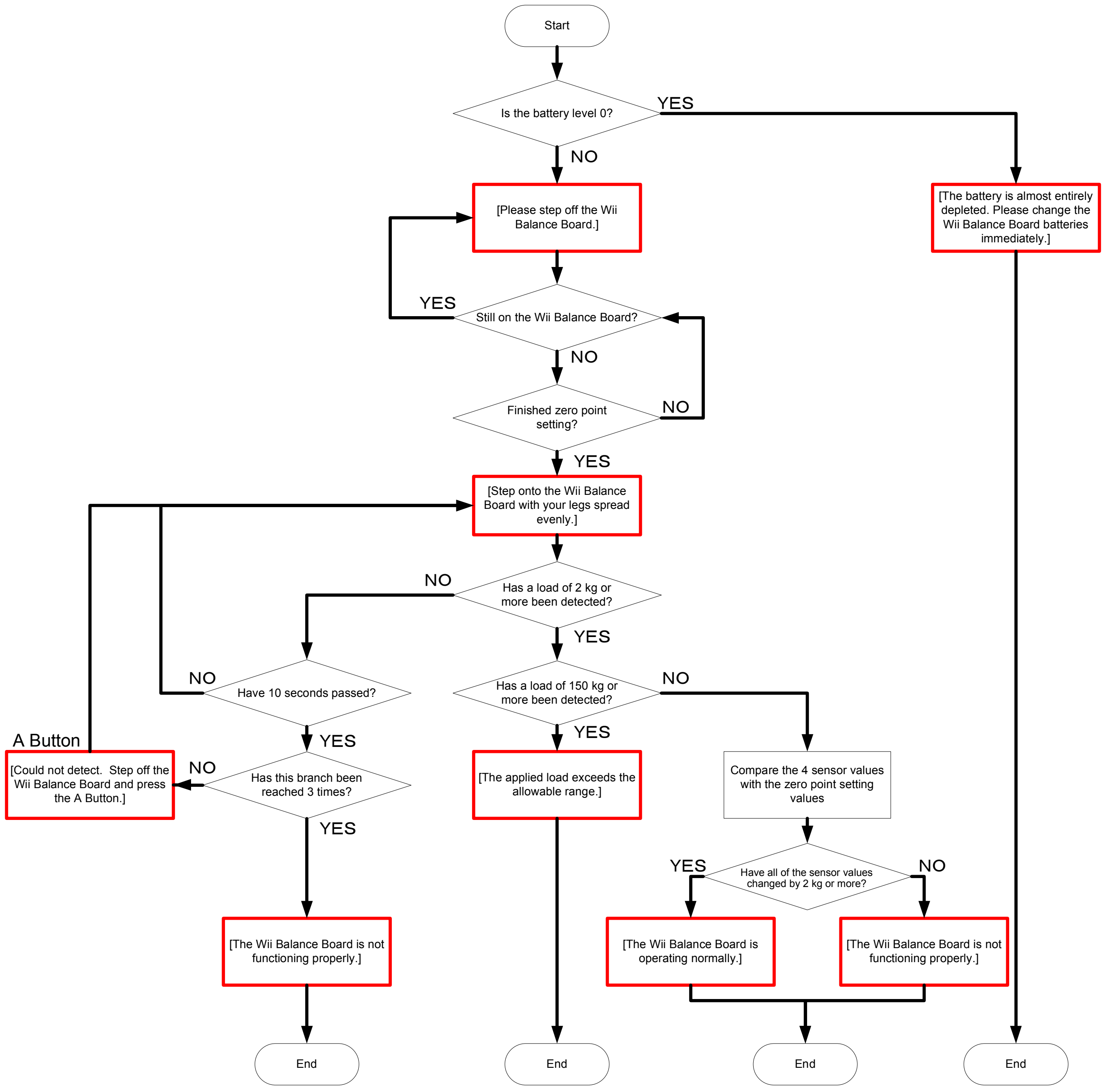
1. Use the `WBCGetBatteryLevel` function to check the battery level; if `0` is returned, prompt the user to replace the batteries.
2. Display "Please step off the balance board." Have the user step off the board, and then verify whether the user has stepped off using the `WBCRead` function's return value.
3. After confirming the user has stepped off, set the zero point.
4. Display "Step onto the Wii Balance Board with your legs spread evenly." and then prompt the user to step on the Wii Balance Board within 10 seconds with his weight evenly distributed. Verify the user has stepped on the Wii Balance Board by confirming that the load has changed by at least two kilograms compared to the zero point.

   If the user has not stepped on the Wii Balance Board within 10 seconds, display "Could not detect. Step off the Wii Balance Board and press the A Button." Once the A Button is pressed, repeat step 4.

   If the user cannot be detected on the third try, display a message stating the Wii Balance Board accessory is not functioning properly.
5. If the user has been detected, verify that the load does not exceed the maximum. If the maximum load has not been exceeded, compute the load's average over two seconds, and then check the amount of change in the load value for each foot. If the amount of change is greater than or equal to two kilograms, the Wii Balance Board accessory is working normally; otherwise, it is not. Display the result of the troubleshooting (for example, "OK" or "NG") to inform the user whether the Wii Balance Board is broken.

Figure 5-1 shows the Troubleshooting Mode flow. User messages to be displayed on the screen are set off in square brackets inside the red boxes.

Figure 5-1 Troubleshooting Mode Flow


Start
Is the battery level 0?
YES
NO
[The battery is almost entirely depleted. Please change the Wii Balance Board batteries immediately.]
[Please step off the Wii Balance Board.]
YES
Still on the Wii Balance Board?
NO
Finished zero point setting?
NO
YES
[Step onto the Wii Balance Board with your legs spread evenly.]
NO
Has a load of 2 kg or more been detected?
YES
NO
Have 10 seconds passed?
Has a load of 150 kg or more been detected?
NO
A Button
YES
YES
[Could not detect. Step off the Wii Balance Board and press the A Button.]
NO
Has this branch been reached 3 times?
[The applied load exceeds the allowable range.]
Compare the 4 sensor values with the zero point setting values
YES
YES
Have all of the sensor values changed by 2 kg or more?
NO
[The Wii Balance Board is not functioning properly.]
[The Wii Balance Board is operating normally.]
[The Wii Balance Board is not functioning properly.]
End
End
End
End

To verify the troubleshooting mode is implemented correctly, prompt a user to step on one of the four corners of the Wii Balance Board accessory, and then confirm that the board fails the check. Repeat this check on all four corners.

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, because 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 that value will be sent continuously as a result.

Use the `WBCGetBatteryLevel` function to check the battery level; if zero is returned, prompt the user to replace the batteries.

6.2 Message Display Conditions for Battery Replacement

When the battery strength is diminished, the battery level may flicker between 0 and 1 due to hardware characteristics. If the application shows a replace battery message when the battery level is 0 and closes the message when the battery level is 1 or greater, then this could cause the message to be displayed and closed whenever the battery level fluctuates by a small amount. If this seems to be unattractive from a game design viewpoint, try using a battery value of 110 or greater for the condition to close the battery replacement message.