Wii Save Data Creation Guidelines

December 7, 2006

Table of Contents

Code

Figures

Tables

Revision History

Revision Date	Description
12/7/2006	Added 2.2.5 Save Data Permissions.
11/27/2006	Revised the description of the time at which applications should create save banner files in "2, Save Data Overview" Note that our original recommendation that "save banner files be created first" is now prohibited.
11/14/2006	Revised "2.1.6, File Names and Location" A restriction has been added for handling a "notransfer" directory. Revised "2.2.4, Limitations."
11/3/2006	Revised "2.1.6, File Names and Location." Be absolutely sure to reread this section as the method for handling the "nocopy" directory has changed.
10/10/2006	Corrected the mismatched definitions of `NAND_BANNER_ICON_ANIM_SPEED_FAST` and `NAND_BANNER_ICON_ANIM_SPEED_SLOW` in "2.1.1 File Format." Added a note regarding alpha image values in "2.1.3 Icons" and "2.1.4 Banner." **This note contains critical information. Be sure to read it.**
9/19/2006	Added to and revised section 2.2, Save Data Specifications.
9/15/2006	Initial version.

1 Introduction

This document describes the specifications and cautions that you should observe when creating save data in Wii applications.

Use the NAND library whenever you create save data. Unlike Nintendo GameCubeTM, the Wii console NAND memory used to store save data is mounted in the Wii console. This makes it is easier to handle than the CARD library.

This document describes the differences between Wii application save data and Nintendo GameCube software save data. Sample code is also provided at the end of this document.

The program code used in this document was based on all the information available at the time this document was written. It may not match the information for the most recent application development environment. In addition, the correct functioning of this code is not guaranteed when it is built and run. See the most recent programming manual and function reference when implementing save data operations in an actual application.

2 Save Data Overview

Wii application save data differs from Nintendo GameCube software save data in the following ways:

- Only images in RGB5A3 format can be used in the icons and banner.
- The texture size of the icons and banner are different. (The icons are 32 x 32, instead of 48 x 48 and the banner is 96 x 32, instead of 192 x 64.)
- The icons, banner, and comments are stored in a file independent of the application-specific save data.
- The banner data cannot be omitted.

Independent of the application-specific save data, the file that stores the information for the icons, banners, and comments is called the save banner file. Applications must create this save banner file immediately after creating active save data.

If there is no save banner file in the save data directory, specifications now call for corresponding save data to be deleted from the Wii Menu as invalid
data. Be sure to create save banner files in the directory so that no save data is lost unexpectedly.

If the save data directory contains only save banner files (with no save data in them), corresponding save data is not deleted, but it becomes impossible to copy that data onto an SD card from the Wii Menu.
To avoid unnecessary confusion for the user, do not create save banner files when there is no valid save data.

2.1 Save Banner File Specifications

The save banner file contains the following:

- A copy protect flag for all save data
- Icon data (minimum of one; maximum of eight)
- Icon animation settings (two types of playback methods and three display speeds for each segment)
- Banner data (one)
- Comment (20 characters x 2 lines)

2.1.1 File Format

The format of the save banner file is defined in Code 2-1.

Code 2-1 Header File Code

```
#define NAND_BANNER_TEXTURE_SIZE (192 * 64 * 2)
#define NAND_BANNER_ICON_SIZE    ( 48 * 48 * 2)
#define NAND_BANNER_COMMENT_SIZE 32

#define NAND_BANNER_ICON_ANIM_SPEED_END    0
#define NAND_BANNER_ICON_ANIM_SPEED_FAST   1
#define NAND_BANNER_ICON_ANIM_SPEED_NORMAL 2
#define NAND_BANNER_ICON_ANIM_SPEED_SLOW   3
#define NAND_BANNER_ICON_ANIM_SPEED_MASK   3

#define NAND_BANNER_FLAG_NOTCOPY 0x00000001

#define NAND_BANNER_FLAG_ANIM_BOUNCE 0x00000010
#define NAND_BANNER_FLAG_ANIM_LOOP   0x00000000
#define NAND_BANNER_FLAG_ANIM_MASK   0x00000010

#define NAND_BANNER_SIGNATURE 0x5749424E

#define NANDGetIconSpeed(stat, n)      (((stat)->iconSpeed  >> (2 * (n))) &
NAND_BANNER_ICON_ANIM_SPEED_MASK)
#define NANDSetIconSpeed(stat, n, f)   ((stat)->iconSpeed  = (u16) (((stat)-
>iconSpeed  & ~(NAND_BANNER_ICON_ANIM_SPEED_MASK << (2 * (n)))) | ((f) << (2 *
(n)))))

#define NAND_BANNER_SIZE( i ) ( 32 + NAND_BANNER_COMMENT_SIZE * sizeof(u16) * 2
+ NAND_BANNER_TEXTURE_SIZE + NAND_BANNER_ICON_SIZE * (i) )

typedef struct
{
    u32 signature;            // Signature: 0x5749424E

    u32 flag;                 // Flags
    u16 iconSpeed;            // Icon animation speed
    u8  reserved[22];         // for 32B align

    u16 comment[2][NAND_BANNER_COMMENT_SIZE];    // title and comment
    u8  bannerTexture[NAND_BANNER_TEXTURE_SIZE]; // Banner texture
    u8  iconTexture[8][NAND_BANNER_ICON_SIZE];   // Icon texture 0-7
} NANDBanner;

void NANDInitBanner( NANDBanner *bnr, u32 flag, const u16 *title, const u16
*comment );
```

The code shown in Code 2-1 is extracted from the portion of the `nand.h` header file that is related to the `NANDBanner` structure.

The save banner file sets the necessary information in the `NANDBanner` structure defined in the header file, and then saves that information as a file.

The structure data members are described in Table 2-1. The contents set with each data member are shown in Table 2-2, Table 2-3, and Figure 2-1.

Table 2-1 Description of NANDBanner Data Members

Size	Name	Contents
4	`signature`	The file identifier for the save banner file (fixed to 0x5749424E).
4	`flag`	Flag types.
2	`iconSpeed`	Specifies the icon animation speed, using 2 bits per icon.
22	`reserved`	Padding.
64	`comment[0]`	First comment line. **Important:** The game title must be provided here.
64	`comment[1]`	Second comment line.
24576	`bannerTexture`	Banner image data in 192 x 64 pixels, RGB5A3 format.
4608	`iconTexture[0]`	Icon image data for the first segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[1]`	Icon image data for the second segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[2]`	Icon image data for the third segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[3]`	Icon image data for the fourth segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[4]`	Icon image data for the fifth segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[5]`	Icon image data for the sixth segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[6]`	Icon image data for the seventh segment in 48 x 48 pixels, RGB5A3 format.
4608	`iconTexture[7]`	Icon image data for the 8th segment in 48 x 48 pixels, RGB5A3 format.

Table 2-2 NANDBanner.flag Values

Definition	Value	Contents
`NAND_BANNER_FLAG_NOTCOPY`	0x00000001	All save data is copy protected.
`NAND_BANNER_FLAG_ANIM_BOUNCE`	0x00000010	Animation playback method is set to bounce.
`NAND_BANNER_FLAG_ANIM_LOOP`	0x00000000	Animation playback method is set to loop.
`NAND_BANNER_FLAG_ANIM_MASK`	0x00000010	Mask for the animation playback method.

Figure 2-1 Contents Set in NANDBanner.iconSpeed

15			8	7			0
Playback speed of the eighth segment	Playback speed of the seventh segment	Playback speed of the sixth segment	Playback speed of the fifth segment	Playback speed of the fourth segment	Playback speed of the third segment	Playback speed of the second segment	Playback speed of the first segment

Table 2-3 Playback Speed Settings

Definition	Value	Playback Speed
`NAND_BANNER_ICON_ANIM_SPEED_END`	0	---
`NAND_BANNER_ICON_ANIM_SPEED_FAST`	1	Fast
`NAND_BANNER_ICON_ANIM_SPEED_NORMAL`	2	Normal
`NAND_BANNER_ICON_ANIM_SPEED_SLOW`	3	Slow

2.1.2 Texture Data

The icons and banners displayed using the Wii System Menu are created as texture data. The texture data format supports only RGB5A3.

Create texture data, using a 4:3 aspect ratio. Make adjustments so that icons and banners are displayed in a ratio that is equivalent to 4:3, even if the aspect ratio of the screen is 16:9.

2.1.3 Icon

The icon texture settings specify how the icon is displayed on the save data selection screen of the Wii system menu.

Icons use one or more textures (up to a maximum of eight) composed of 48 x 48 pixels in RGB5A3 format.

Icons can be animated by using multiple textures. The animation speed varies according to the number of frames displayed in each segment.

Table 2-4 Icon Animation Speed Specifications

Definition	Number of Displayed Frames in Each Segment	Animation Speed
`NAND_BANNER_ICON_ANIM_SPEED_END`	---	---
`NAND_BANNER_ICON_ANIM_SPEED_FAST`	4	Fast
`NAND_BANNER_ICON_ANIM_SPEED_NORMAL`	8	Normal
`NAND_BANNER_ICON_ANIM_SPEED_SLOW`	12	Slow

If the number of displayed segments (number of icons) is less than eight, the display speed of the segment that follows the last displayed segment is set to `NAND_BANNER_ICON_ANIM_SPEED_END`.

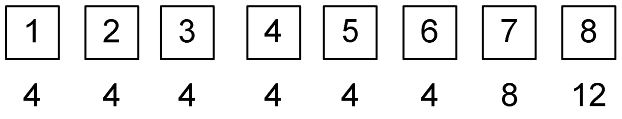
For example, if the number of displayed segments is 4, the fifth segment is set to `NAND_BANNER_ICON_ANIM_SPEED_END`.

Code 2-2 Code Example for Four Displayed Segments

```
NANDSetIconSpeed( &info, 0, NAND_BANNER_ICON_ANIM_SPEED_SLOW );
NANDSetIconSpeed( &info, 1, NAND_BANNER_ICON_ANIM_SPEED_NORMAL );
NANDSetIconSpeed( &info, 2, NAND_BANNER_ICON_ANIM_SPEED_FAST );
NANDSetIconSpeed( &info, 3, NAND_BANNER_ICON_ANIM_SPEED_SLOW );
NANDSetIconSpeed( &info, 4, NAND_BANNER_ICON_ANIM_SPEED_END );
```

If the animation has eight segments, texture data for eight segments is created as described below. (The numbers shown in Figure 2-2 indicate the number of frames displayed in each of the eight segments shown. Each frame has a duration of 1/60 second.)

Figure 2-2 Number of Displayed Frames Per Segment



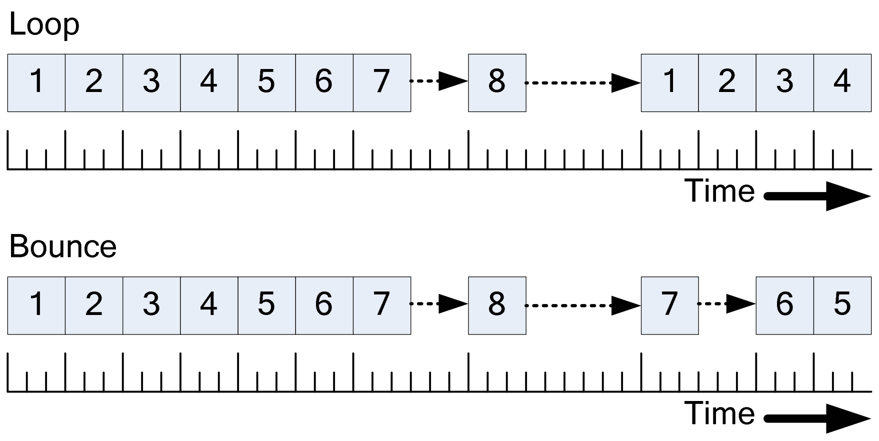
Select either loop playback or bounce playback as the animation playback method.

Table 2-5 Icon Animation Speed Method

Definition	Playback Method
`NAND_BANNER_FLAG_ANIM_LOOP`	Loop playback
`NAND_BANNER_FLAG_ANIM_BOUNCE`	Bounce playback

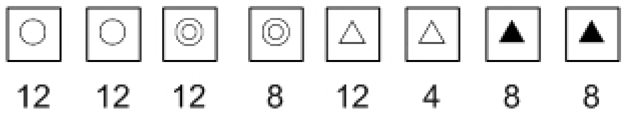
The animation is played as shown in Figure 2-3, according to the specified animation playback method. (In the time gauge, each tick mark represents 1/60 second.)

Figure 2-3 Loop and Bounce Animation Playback Methods



To show the same segment for more than 13 frames, apply the same texture to multiple segments.

Figure 2-4 Example Animation with Frame Counts



With the settings shown in Figure 2-4, ○ is displayed for 24 frames, ◎ for 20 frames, △ for 16 frames, and ▲ for 16 frames.

Note: The alpha image values of icons are correctly reflected on any Wii console.

2.1.4 Banner

The bannerTexture setting saves the banner displayed on the save data detail screen of the Wii system menu.

Use one texture composed of 192 x 64 pixels in RGB5A3 format.

Note: The alpha image values of banners are not correctly reflected on some Wii consoles for North America. Therefore, do not use alpha image values for banners in applications marketed in North America.

2.1.5 Comment

The `comment` setting saves the comment displayed on the save data detail screen, using the Wii System Menu.

Put the game title on the first line. The second line can have any sort of comment. If the comment does not fit on the second line, the first line can contain some of the text, after the game title.

Comment characters are encoded in Unicode. The NULL character is treated as a terminating character for the string. However, to prevent unnecessary characters being left over after the comment, fill the rest of the line with NULLs.

The Gothic font (WBF1) of the WBF (Wii Bitmap Font) can be used to display the comment. All the characters except the extended characters in WBF1 can be used. (See Table 2-6 Characters That Can Be Used in the Comment).

The structure can store 32 characters for each line. However, only 20 characters are guaranteed to be displayed. Fill from the twenty-first character and beyond with NULLs.

Table 2-6 Characters That Can Be Used in the Comment

Usable Characters
English characters (Latin characters)
Greek characters
JIS single-byte characters
JIS non-Kanji
JIS Level-1 Kanji set
JIS Level-2 Kanji set
Alphanumeric characters

2.1.6 File Names and Location

Create the save banner file with the name `banner.bin` in the directory following directory:

`/title/`<title-id(Hi)>`/`<title-id(Low)>`/data`

This directory is created for save data by the system.

> Only one save banner file can be created by the application. Create it in the designated directory for save data with the specified file name.

Because the directories named `noerase` and `notransfer` are reserved by the system, creation of these directories in a directory used for saving data is prohibited.

In addition, since files from the directory created under the name `nocopy` can be restricted from copying to an SD Card, you can use this directory when creating save data for temporary disconnection. However, do not create data that may have a fatal impact on game progress if lost in the `nocopy` directory.

If this type of data is saved in the `nocopy` directory, original data cannot be restored if the user erases data in NAND memory with the intention of having made a save data backup on the SD Card.

2.1.7 File Size

The size of the save banner file is calculated with the `NAND_BANNER_SIZE` macro.

```
Minimum file size for the save banner file (one icon) =
NAND_BANNER_SIZE( 1 ) =
32 (header) + 32 x 2 x 2 (comment) + 192 x 64 x 2 (banner) + 48 x 48 x 2 x 1
(icon) =
32 + 128 + 24,576 + 4,608 = 29,344 bytes = 28.65625 kilobytes

Maximum file size for the save banner file (eight icons) =
NAND_BANNER_SIZE( 8 ) ＝
32 (header) + 32 x 2 x 2 (comment) + 192 x 64 x 2 (banner) + 48 x 48 x 2 x 8
(icon) =
32 + 128 + 24,576 + 36,864 = 61,600 bytes = 60.15625 kilobytes
```

To save the save banner file, a minimum of 28.7 KB and a maximum of 60.2 KB of storage is required. Because the NAND library file block size is 16 kilobytes, the minimum and maximum are set to 32 KB and 64 KB, respectively, for exclusive use in the file system.

A comparison of the number of icons and the size of the save banner file is summarized in Table 2-7.

Table 2-7 Number of Icons and Size of the Save Banner File

Number of Icons	File Size (bytes)	File Size (KB)	Exclusive Use Size (KB)
1	29,344	28.65625	32
2	33,952	33.15625	48
3	38,560	37.65625	48
4	43,168	42.15625	48
5	47,776	46.65625	48
6	52,384	51.15625	64
7	56,992	55.65625	64
8	61,600	60.15625	64

2.2 Save Data Specifications

By separating the information displayed using the Wii System Menu into the save banner file, the application can freely design the layout of the file created as save data.

2.2.1 Number of Blocks

The size of a single block for the save data file is 128 KB. Caution is required when calculating the number of blocks needed for the save data to be displayed by the application because this value is different from the file block size (16 KB) used by the NAND library.

2.2.2 Precautions When Creating Save Data

Be careful when using the NAND library `NANDSafe` series functions to create save data or when moving the file after creating a temporary file in order to prevent the important user save data from being corrupted or lost.

See the NAND library Function Reference for cautions regarding the use of the `NANDSafe` series functions.

2.2.3 Anti-Falsification Processing

Anti-falsification processes are executed when save data is copied to an SD Card, using the Wii System Menu. Therefore, applications do not have to implement their own anti-falsification processing when storing save data on the Wii console.

2.2.4 Limitations

Because the save banner file must be created when save data is created, the number of files that the application can create in Wii console NAND memory is limited to the amount needed for the save banner file.

See the NAND library Function Reference or the section in the *Wii Programming Guidelines* that discusses Wii console NAND memory for limitations on the number of save data, file names, or file sizes.

In addition, you must not save information that applies only to specific Wii consoles, such as MAC addresses, in the save data. This prevents data from becoming unusable if the Wii console or its modules require replacement during servicing.

2.2.5 Save Data Permissions

To ensure that save data is handled correctly by the Data Management feature in the Wii Menu, be sure to give the owner read permission to all directories and files in the save data directory.

Specifically, do not create any files or directories with no read permission for the owner or perform any permission modification that would remove owner read permission from an existing file or directory.

3 Creating Save Data

An example of creating a save banner file is shown in this chapter with sample code.

3.1 Setting Information in the NANDBanner Structure

The example in Code 3-1 assumes that the banner image data has already been loaded in the banner variable and that the icon image data has been loaded in the `icon0` to `icon3` variables from a file.

Code 3-1 Setting Information in the NANDBanner Structure

```
NANDBanner info ATTRIBUTE_ALIGN(32);

// Initialization of the NANDBanner Structure
NANDInitBanner( &info,
                NAND_BANNER_FLAG_NOTCOPY,
                L "Game Title",
                L "Comment" );   //When all save data is copy protected

std::memcpy( info.bannerTexture, banner, NAND_BANNER_TEXTURE_SIZE ); // Banner
texture
std::memcpy( info.iconTexture[0], icon0, NAND_BANNER_ICON_SIZE );
std::memcpy( info.iconTexture[1], icon1, NAND_BANNER_ICON_SIZE );
std::memcpy( info.iconTexture[2], icon2, NAND_BANNER_ICON_SIZE );
std::memcpy( info.iconTexture[3], icon3, NAND_BANNER_ICON_SIZE );
NANDSetIconSpeed( &info, 0, NAND_BANNER_ICON_ANIM_SPEED_SLOW );
NANDSetIconSpeed( &info, 1, NAND_BANNER_ICON_ANIM_SPEED_NORMAL );
NANDSetIconSpeed( &info, 2, NAND_BANNER_ICON_ANIM_SPEED_FAST );
NANDSetIconSpeed( &info, 3, NAND_BANNER_ICON_ANIM_SPEED_SLOW );
NANDSetIconSpeed( &info, 4, NAND_BANNER_ICON_ANIM_SPEED_END ); // When there are
less than eight, enter END
info.flag |= NAND_BANNER_FLAG_ANIM_BOUNCE; // For bounce animation
```

3.2 Creating Files Using NANDSafe Series Functions

The structure described in section 3.1 is created using the NANDSafe line of functions in the `banner.bin` file in the home directory, using the `NANDSafe` series functions.

Code 3-2 Creating Files Using NANDSafe Series Functions

```
#define COPYBUF_SIZE 4096

NANDFileInfo       fileInfo;
s32                  rv;
u32                  len;
char                 homePath[NAND_MAX_PATH];
s8                   copyBuf[COPYBUF_SIZE] ATTRIBUTE_ALIGN(32);

// The content of info is written to the file named banner.bin using the NAND
API
// Obtains the home directory
rv = NANDGetHomeDir(homePath);
if (rv != NAND_RESULT_OK)
{
    OSReport("NANDGetHomeDir failed %d\n", rv);
    return;
}
// Moves to the home directory
rv = NANDChangeDir(homePath);
if (rv != NAND_RESULT_OK)
{
    OSReport("NANDChangeDir failed %d\n", rv);
    return;
}

rv = NANDCreate("banner.bin", NAND PERM OWNER READ | NAND PERM OWNER WRITE |
NAND_PERM_GROUP_READ, 0);
if (rv != NAND_RESULT_EXISTS && rv != NAND_RESULT_OK)
{
    OSReport("NANDCreate failed %d\n", rv);
    return;
}
rv = NANDSafeOpen("banner.bin", &fileInfo, NAND_ACCESS_WRITE, copyBuf,
COPYBUF_SIZE);
if (rv != NAND_RESULT_OK)
{
    OSReport("NANDSafeOpen: Could not open 'banner.bin' for writing\n");
    return;
}
len = NAND_BANNER_SIZE( 4 );    // Four icons are used
rv = NANDWrite(&fileInfo, &info, len);
if (rv != len)
{
    OSReport("NANDWrite: Could not write rv=%d len=%d\n", rv, len);
}

NANDSafeClose(&fileInfo);
```