Wii Save Data Specifications

2010/05/18

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

RVL-06-0175-001-I
Released: August 25, 2010

Table of Contents

Code

Figures

Tables

Revision History

Revision Date	Description
2010/05/18	Changed title of section 3.2 from Creating Files Using NANDSimpleSafe Series Functions to Creating the File Using the /tmp Directory. Updated the sample code to reflect the changes.
2010/01/25	Deleted section 2.2.5 Save Data Permissions. The Revolution SDK that required this restriction is an old version.
2009/11/24	Added a reference to Chapter 8 Wii Console NAND Memory in the *Wii Programming Guidelines* in chapter 22. Save Data Overview. Deleted a sentence (“Each frame has a duration of 1/60 second”) from section 2.1.3 IconIcon because 1/60 second is an NTSC specification. In PAL50, it is 1/50 second. Revised the description in section 2.1.5 CommentComment to refer to the *Wii Programming Guidelines*. Revised the description in section 2.1.6 Filenames and Location to refer to the *Wii Programming Guidelines*. Revised the description in section 2.2.2 Precautions When Creating Save Data to refer to the *Wii Programming Guidelines*. Revised the description in section 2.2.3 Anti-Falsification Processing to refer to the *Wii Programming Guidelines*. Revised the description in section 2.2.5 Save Data Permissions to refer to the *Wii Programming Guidelines*. Revised the description related to NANDSafe series functions in section 3.2 Creating the File Using the /tmp Directory and changed description to `NANDSimpleSafe` series functions
2008/05/16	Replaced NANDSafe series functions with NANDSimpleSafe series functions in sections 2.2.2 Precautions When Creating Save Data and 3.2 Creating Files Using NANDSimpleSafe Series Functions. Added additional information about deleting data in section 4.3 Format Wii System Memory.
2008/05/09	Revised an explanation in Chapter 2 Save Data Overview regarding times at which save data will be treated as invalid and deleted because the timing of the deletion was not described accurately. Deleted warning notes related to alpha image values in section 2.1.3 Icon and section 2.1.4 Banner. Along with updates to the Wii Menu, all Wii consoles will now apply alpha image values for banners correctly.
2006/12/07	Added section 2.2.5 Save Data Permissions.
2006/11/27	Revised the description of the time at which applications should create save banner files in Chapter 2 Save Data Overview. Note that our original recommendation that “save banner files be created first” is now prohibited.
2006/11/14	Revised section 2.1.6 Filenames and Location. A restriction has been added for handling a `notransfer` directory. Revised section 2.2.4 Limitations.
2006/11/03	Revised section 2.1.6 Filenames and Location. Be absolutely sure to reread this section as the method for handling the “nocopy” directory has changed.
2006/10/10	Corrected the mismatched definitions of `NAND_BANNER_ICON_ANIM_SPEED_FAST` and `NAND_BANNER_ICON_ANIM_SPEED_SLOW` in section 2.1.1 File Format. Added a note regarding alpha image values in sections 2.1.3 Icons and 2.1.4 Banner. **This note contains critical information. Be sure to read it.**
2006/09/19	Added to and revised section 2.2 Save Data Specifications.
2006/09/15	Initial version.

1 Introduction

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

In this document, save banner files and save data specifications are explained in the overview of save data, and sample code for creating save data is provided at the end.

Note: 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 32x32 pixels, instead of 48x48; and the banner is 96x32, instead of 192x64.)
- 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.

Save banner files and save data specifications described in this document are required items in the *Wii Programming Guidelines*. For details, see chapter 8 Wii Console NAND Memory in the *Wii Programming Guidelines*.

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 as 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 two 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 192x64 pixels, RGB5A3 format.
4608	iconTexture[0]	Icon image data for the first segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[1]	Icon image data for the second segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[2]	Icon image data for the third segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[3]	Icon image data for the fourth segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[4]	Icon image data for the fifth segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[5]	Icon image data for the sixth segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[6]	Icon image data for the seventh segment in 48x48 pixels, RGB5A3 format.
4608	iconTexture[7]	Icon image data for the 8th segment in 48x48 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

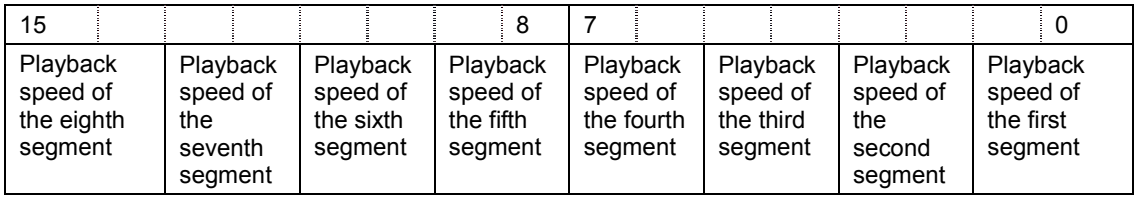


Table 2-3 Playback Speed Settings

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 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 Menu.

Icons use one or more textures (up to a maximum of eight) composed of 48x48 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`.

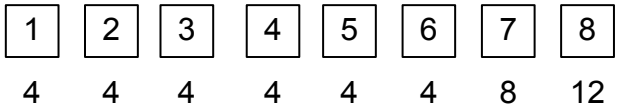
For example, if the number of displayed segments is 4, the fifth segment is set to `NAND_BANNER_ICON_ANIM_SPEED_END`.

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.)

Figure 2-2 Number of Displayed Frames Per Segment



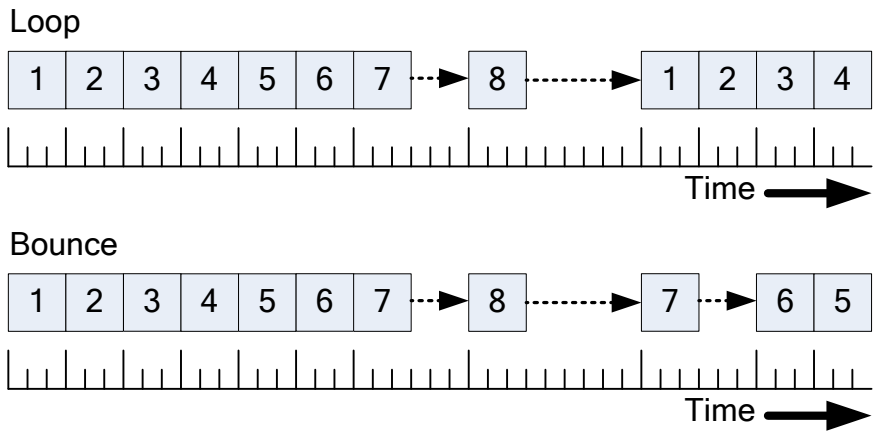
Select either loop playback or bounce playback as the animation playback method.

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.

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.

2.1.4 Banner

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

Use one texture composed of 192x64 pixels in RGB5A3 format.

2.1.5 Comment

For details, see section 8.20 Save Data Comments [Required] in the *Wii Programming Guidelines*.

2.1.6 Filenames and Location

For details, see sections 8.21 Save Data Location Specification [Required], 8.22 Prohibition of Using Reserved Names [Required], and 8.23 Precautions Regarding the nocopy Directory [Required] in the *Wii Programming Guidelines*.

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 KB

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 KB
```

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 KB, 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 2-6.

Table 2-6 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

For details, see section 8.19 Precautions for Creating Save Data [Information] in the *Wii Programming Guidelines*.

2.2.3 Anti-Falsification Processing

For details, see section 8.24 Save Data Tampering Protection by the Wii Console [Information] in the *Wii Programming Guidelines*.

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 section 8.1 Restrictions on Wii Console NAND Memory [Required] in the *Wii Programming Guidelines* that discusses limitations on the number of save data, file names, or file sizes.

In addition, do not save information that applies only to specific Wii consoles, such as MAC addresses, in the save data. Please refer to section 8.25 Prohibition of Usage Restrictions on Save Data Due to Wii Console-Specific Information [Required] in the Wii Programming Guidelines.

3 Creating Save Data

This chapter describes how to create a save banner file and provides 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 the File Using the /tmp Directory

The structure created in section 3.1 is used to create a file called `banner.bin` in the home directory. The /tmp directory is used as temporary file storage for this process.

Code 3-2 Creating Files Using /tmp

```
#define COPYBUF_SIZE NAND FSBLOCK SIZE

NANDFileInfo       fileInfo;
s32                  rv;
u32                  len;
char                 homePath[NAND_MAX_PATH];
static u8            copyBuf[COPYBUF_SIZE] ATTRIBUTE_ALIGN(32);

// The sample 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 /tmp
rv = NANDChangeDir("/tmp");
if (rv != NAND_RESULT_OK)
{
    OSReport("NANDChangeDir failed %d\n", rv);
    return;
}

// Create save banner file in /tmp
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 = NANDOpen("banner.bin", &fileInfo, NAND_ACCESS_WRITE, copyBuf, COPYBUF_SIZE);
if (rv != NAND_RESULT_OK)
{
    OSReport("NANDOpen: 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);
}

rv = NANDClose(&fileInfo);
if (rv != NAND_Result_OK)
{
    OSReport("NANDClose: Could not close 'banner.bin' %d\n", rv);
    return;
}
// Move save banner file to home directory.
```

```
rv = NANDMove("banner.bin", homePath);
if (rv != NAND_Result_OK)
{
    OSReport("NANDMove: Could not move 'banner.bin' to homePath %d\n", rv);
    return;
}
```