

TWL-SDK

TWL-SDK Migration Guide

Precautions When Switching from NITRO-SDK

2009/05/20

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

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and/or its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Introduction	6
2	Notes Applicable to All Applications	7
2.1	Revision to the NITROSDK_ROOT Environment Variable	7
2.2	Added OS_CONSOLE_TWL to the Return Values of OS_GetConsoleType	7
2.3	Execution Stops on OS_Panic When Going Into Sleep Without Stopping Wireless Communications	7
2.4	Increase in Amount of Memory Required to Link the Debugger Library	7
2.5	Increase in Processing Time of Debug Output	7
2.6	Deletion of the WC Sample Library	7
2.7	Increased Speed of STD Functions	8
2.8	Elimination of Support for %n Format with STD_TSprintf	8
2.9	Bug Fix for MATH_FFTRReal and MATH_IFFTRRead	8
2.10	Deletion of SVC_IsMmemExpanded	8
2.11	MATH MD5 Function Not Recommended	8
3	Notes When Porting to NITRO ROM	9
3.1	Increase in Amount of Memory Used by the Library	9
4	Notes on HYBRID and LIMITED ROM Commonalities	10
4.1	Reduced ROM Access Speed in TWL Mode (Card Applications)	10
4.2	Necessity of Calling OS_EnableIrq Before FS_Init	10
4.3	Addition of libsyscall.twl.a and rom_header.LTD.sbin	10
4.4	Revised Start Address of the Static Module	10
4.5	Change from NTR Banners to TWL Banners	10
4.6	About DS Download Play Child Devices	11
4.7	Getting the Wireless Communications ON/OFF Setting	11
4.8	Change to ARM7 Component ichneumon	11
4.9	Notes on Using OS_SpinWait	11
4.10	Parental Controls Settings	11
4.11	About the CTRDG and VIB Libraries	11
4.12	RSF File Revisions	12
4.12.1	Revision to RomSpeedType (Card Applications)	12
4.12.2	WramMapping Setting	12
4.12.3	Codec Mode Setting	12
4.12.4	CardRegion Setting (Card Applications)	12
5	Notes on Porting to HYBRID ROM	13
5.1	Increase in Amount of Memory Used by the Library	13
5.1.1	Disabling the Cartridge Library (CTRDG)	13

5.1.2	Using the Beginning Memory Region	13
6	Notes on Porting to LIMITED ROM	14
6.1	No Support for Clone Boots	14
7	Notes on Porting to a NAND Application	15
7.1	Linking the NA Library	15
7.2	Using OS_RebootSystem in Place of OS_ResetSystem.....	15
7.3	Return Value of OS_GetConsoleType.....	15
7.4	Provision of an Electronic Manual	15
7.5	About Save Data	15
7.5.1	Deleting CARD Functions.....	15
7.5.2	Use of FS Functions	16
7.5.3	User Initialization of Private Save Data	16
7.5.4	About the Size Specified for Save Data.....	16
7.5.5	Improved Access Speed Compared to Flash Memory	16
7.5.6	Power Shutoff During Save	16
7.5.7	Considering the NAND Write Lifetime	16
7.5.8	New FS Errors Requiring Handling.....	17
7.6	RSF File Revisions	17
7.6.1	InitialCode Setting.....	17
7.6.2	Size Specification for Private Save Data and Public Save Data.....	17
7.6.3	Revising RomSize and RomFootPadding	17
7.7	Notes on Using FS Functions to Access ROM Archives	17
7.7.1	Addition of Random Delay to NAND Read Times	18
7.7.2	Higher Speed Access of NAND Application ROM Archives.....	18
7.7.3	Inability to Set a DMA Channel with FS_Init	18
7.7.4	Delayed RTC, SNDEX, and Wireless Processing	18
7.7.5	Reduced File Access Speed Using MP Communications	18

Revision History

Date	Description
2009/05/20	Changed "TWL-SDK 5.1" to "TWL-SDK 5.2" as the comparative subject for TWL-SDK. Added descriptions of the <code>OS_CreateExtraHeap</code> and <code>OS_AddExtraAreaToHeap</code> functions in section 5.1.2 Using the Beginning Memory Region.
2009/03/10	Changed "TWL-SDK 5.1 PR" and "TWL-SDK 5.1 RC" to "TWL-SDK 5.1" as the comparative subject for TWL-SDK. Deleted text relating to Wi-Fi from sections 3.1 and 5.1. Added an explanation about the internal behavior of <code>WM_Initialize</code> to section 4.7. Added to the explanation about <code>CodecMode</code> in section 4.12.3 Added the section 4.12.4 <code>CardRegion</code> Settings. Deleted section 5.2 About Clone Boot. Added section 7.6.3 <code>RomSize</code> and <code>RomFootPadding</code> .
2009/02/23	Added section 2.11 MATH MD5 Function Not Recommended. Changed the amount of increased memory to 15 KB when transplanting in HYBRID ROM in section 5.1. Added section 5.1.1 Disabling the Cartridge Library (CTRDG). Added section 5.1.2 Using the Beginning Memory Region.
2009/02/06	Added section 3.1 Increase in Amount of Memory Used by the Library. Revised section 5.1 Increase of Memory Used by the Library.
2009/01/07	Changed values specified in section 4.12.2 <code>WramMapping</code> Setting.
2008/12/18	Deleted section 3.1 Temporary Change to the Start Address of the Static Module. Moved the content of section 7.7.1 Addition of FS Errors to section 7.5.8 New FS Errors. Deleted section 7.7.5 Increased Speed Due to Memory Alignment When Using <code>FS_ReadFile</code> . Deleted section 7.7.6 Reduced Speed of <code>FS_ReadDirectory</code> . Deleted section 7.7.7 Deletion of Functions That Use File IDs.
2008/11/17	Initial version.

1 Introduction

This document is intended for developers who have been using the NITRO-SDK. It covers topics that require attention when migrating to the TWL-SDK.

The purpose of this document is to describe how to take an existing NITRO-SDK project and rebuild it for the TWL-SDK so that it runs. Because the purpose is to just get applications running, no new features will be introduced in this document. Only topics where changes are required to migrate to the TWL-SDK are discussed.

The comparisons given are made between the NITRO-SDK 4.2 plus 8 patch and TWL-SDK 5.2.

This document includes notes on the various different ROM types available, referred to as NITRO, HYBRID, and LIMITED. For details on ROM types, see the [TWL-SDK Application Development Guide](#).

2 Notes Applicable to All Applications

This section covers common points to be noted when migrating to any of the NITRO, HYBRID, and LIMITED ROM types.

2.1 Revision to the NITROSDK_ROOT Environment Variable

The `NITROSDK_ROOT` environment variable used under NITRO-SDK has been changed to `TWLSDK_ROOT` under TWL-SDK. The platform must be newly set to `TWLSDK_PLATFORM`. For details, see the *Quick Start Guide*.

2.2 Added `OS_CONSOLE_TWL` to the Return Values of `OS_GetConsoleType`

`OS_CONSOLE_TWL` has been added to the return values of the `OS_GetConsoleType` function under TWL-SDK. `OS_CONSOLE_TWL` returns for applications running in TWL mode on actual TWL hardware. For applications running in NITRO mode on actual TWL hardware, `OS_CONSOLE_TWL` returns only when in DEBUG or RELEASE mode, but `OS_CONSOLE_NITRO` returns for FINALROM builds.

2.3 Execution Stops on `OS_Panic` When Going Into Sleep Without Stopping Wireless Communications

Heretofore, applications used to run under NITRO-SDK when entering sleep mode without stopping wireless operations. Now, under TWL-SDK, execution is stopped by an `OS_Panic` function call. This was done to support guidelines that prohibited wireless operations when transitioning to sleep mode.

2.4 Increase in Amount of Memory Required to Link the Debugger Library

If both the IS-TWL-DEBUGGER software and the IS-NITRO-DEBUGGER software have been installed, the debugger libraries for both will be linked, leaving less memory available for use by the application.

2.5 Increase in Processing Time of Debug Output

The time required to output debug information with functions such as `OS_Printf` when using IS-TWL-DEBUGGER is approximately double that compared to using IS-NITRO-DEBUGGER.

2.6 Deletion of the WC Sample Library

The entire WC library used in sample demos for the WM, MB, and WBT libraries has been changed to the WH library. Along with this move, the WC library has been deleted.

2.7 Increased Speed of STD Functions

The execution speed of the `STD_GetStringLength`, `STD_CopyLString`, `STD_CompareString`, and `STD_CompareNString` functions has been increased.

2.8 Elimination of Support for %n Format with STD_TSPrintf

Due to a security problem, support for the `%n` format used by the `STD_TSNPrintf`, `STD_TSPrintf`, `STD_TVSNPrintf`, and `STD_TVSPrintf` functions has been eliminated.

2.9 Bug Fix for MATH_FFTRReal and MATH_IFFTRRead

Although there were problems under NITRO-SDK with the `MATH_FFTRRead` and `MATH_IFFTRRead` functions, these problems have been corrected under TWL-SDK.

2.10 Deletion of SVC_IsMmemExpanded

The `SVC_IsMmemExpanded` function included with NITRO-SDK has been deleted under TWL-SDK. To get the size of main memory, use the `OS_GetConsoleType` function.

2.11 MATH MD5 Function Not Recommended

Of the MATH hash functions included in NITRO-SDK, MD5 is now not recommended because it is not secure. For this reason, it has been deleted from the TWL-SDK function reference. If a hash function is required, use SHA-1. See **MATH (Numerical Computations) → Overview → Hash Message Digest** in the *Function Reference Manual* for details. The MD5 header and function are retained for compatibility.

3 Notes When Porting to NITRO ROM

The notes in this section apply only to when creating a NITRO ROM build under TWL-SDK.

3.1 Increase in Amount of Memory Used by the Library

The memory consumed by the TWL-SDK libraries has increased in comparison to NITRO-SDK due to feature additions and bug fixes. For this reason, when an application is ported from NITRO-SDK, the memory region that the application can use decreases. As an example, when past DS projects are ported to TWL-SDK 5.2, the memory consumed by the library increases by roughly 9 KB.

If a ported application has all NITRO-SDK functions linked, library memory consumption increases by about 11 KB. An increase of about 7 KB is experienced when using Thumb code. However, in cases where the number of inline functions in the library has increased, the actual amount of memory consumed may be larger than these figures because the memory used increases as the number of duplicate calls increases.

4 Notes on HYBRID and LIMITED ROM Commonalities

This section covers common points when porting to HYBRID or LIMITED ROM when using TWL-SDK.

4.1 Reduced ROM Access Speed in TWL Mode (Card Applications)

With card applications, the ROM access speed in TWL mode is reduced as compared to ROM access speed in NITRO mode. Note that, particularly in the case of HYBRID applications, this can lead to different application behavior when running under TWL mode as compared to NITRO mode. For details, see **FS (File System) → Overview → ROM Archives** in the *Function Reference Manual*.

4.2 Necessity of Calling OS_EnableIrq Before FS_Init

When running in TWL mode, the `OS_EnableIrq` function must be called before the `FS_Init` function. If it is not called, the `OS_TPanic` function stops execution inside `FS_Init`. `OS_EnableIrq` does not need to be called first when running a NITRO ROM or HYBRID ROM application in NITRO mode.

4.3 Addition of libsyscall.twl.a and rom_header.LTD.sbin

Nintendo distributes `libsyscall.a` and `rom_header.template.sbin` for each NITRO application, and this same support is provided for TWL applications. You must add `libsyscall.twl.a` and `rom_header.LTD.sbin` to your TWL applications. For these files, use `$TWLSDK/lib/ARM9-TS/etc/libsyscall.twl.a` and `$TWLSDK/tools/bin/rom_header.LTD.sbin`, included in TWL-SDK.

4.4 Revised Start Address of the Static Module

The start address of the Static module has been changed to `0x02004000`. Because this first 16-KB area of memory is used as a system parameter area, it cannot be used by the developer. Developers should note that the amount of available arena memory is reduced by 16 KB in the same way when running HYBRID applications in NITRO mode. The LSF files of some sample demos in NITRO-SDK are set at a static address `0x02000000`, so if you are going to use these files, you need to revise the address to `0x02004000` or `$(ADDRESS_STATIC)` when porting them over.

4.5 Change from NTR Banners to TWL Banners

The banners used under NITRO are called NTR banners, and these banners must be recreated as TWL banners for use with TWL-compatible applications. Specifically, these banners must be remade with the `makebanner.TWL` tool rather than the `makebanner` tool. Banners do not necessarily need to be animated. For details, see **Tools → Overview → Banners** in the *Function Reference Manual*.

4.6 About DS Download Play Child Devices

Only NITRO ROM applications can be run as DS Download Play children. HYBRID ROM and LIMITED ROM versions cannot boot as children. Any ROM running in any mode can be used for a DS Download Play parent device.

4.7 Getting the Wireless Communications ON/OFF Setting

The TWL system's wireless communications feature can be turned off. When this is done, applications that run the `WM_Initialize` function return `WM_ERRCODE_WM_DISABLE` and fail. Inside the `WM_Initialize` function, `WM_Init` returns `WM_ERRCODE_WM_DISABLE` and fails. If you want the user to change the wireless setting, get the Wireless Communications setting of the system with the `OS_IsAvailableWireless` function and then transit to the Wireless Communications option from the System Settings with the `OS_JumpToWirelessSetting` function. Note, however, that this mode transition cannot be made in the case of a HYBRID application running in NITRO mode. If the Wireless Communications setting is turned off and the TWL system is running in NITRO mode, `WM_Initialize` and other wireless library functions will not return errors, but because the wireless communications feature would not run, the system will not be able to find communications partners.

4.8 Change to ARM7 Component `ichneumon`

In the case of HYBRID and LIMITED applications, nothing equivalent to the `ichneumon` component, which has wireless code in VRAM as available for NITRO applications, is provided. For HYBRID applications, use the `mongoose` component, and for LIMITED applications, use the `racoon` component.

4.9 Notes on Using `OS_SpinWait`

Be sure to note that the clock runs twice as fast in TWL mode as in NITRO mode when using the `OS_SpinWait` function. The time required to return from functions differs depending on which mode is running, especially for HYBRID applications. If you want to wait the same amount of time in both modes, use the `OS_SpinWaitSysCycles` function.

4.10 Parental Controls Settings

To create a master ROM, you must set Parental Controls in the SRL file using `MasterEditorTWL`. For details, see the *TWL Master Editor* manual.

4.11 About the CTRDG and VIB Libraries

As for Game Pak (CTRDG) and Rumble Pak (VIB) functions that do not work with TWL systems, they may be linked by HYBRID and LIMITED applications without any problem. On a TWL system, these functions behave as if the Game Pak is not inserted.

4.12 RSF File Revisions

For card applications, make RSF file settings for `TitleName`, `MakerCode`, `RemasterVersion`, `RomSize`, `RomFootPadding`, `RomHeaderTemplate`, and `BannerFile` based on `$TwlSDK/include/twl/specfiles/ROM-TS.rsfc`. Only relevant points are covered here. For details, see **Tools** → **Tools Related to ROM Images** → **makerom.TWL** in the *Function Reference Manual*.

4.12.1 Revision to RomSpeedType (Card Applications)

When a mask ROM (MROM) has been used for `RomSpeedType` with an existing project, be sure to change it to a one-time PROM (1TROM) because MROM cannot be used with HYBRID or LIMITED applications. Pay close attention to application operations because ROM access speed will be reduced due to this change.

4.12.2 WramMapping Setting

`WramMapping` must be set to `MAP2_TS_HYB` for HYBRID applications and to `MAP2_TS_LTD` for LIMITED applications.

4.12.3 Codec Mode Setting

The TWL system has built-in CODEC hardware to control microphone input, speaker output, touchscreen input, and so on. Configure the RSF file with CODEC for operating in TWL mode. If set to NTR (CODEC-DS mode), the CODEC is compatible with NITRO, but the cameras become unusable. To use the cameras, you must configure the setting to TWL (CODEC-TWL mode), but in doing so, you lose compatibility with NITRO ROM and may need to revise the program.

For details on CODEC-TWL mode, see **Sound** → **Overview** → **TWL Extended Sound Features (Overview)** in the *Function Reference Manual*.

4.12.4 CardRegion Setting (Card Applications)

This specifies the application's region. Be aware that the application will not be recognized by the launcher if the application's region is not the same as the TWL system region. Also note that the `ALL` specification in the sample demos is a setting available only for the demos and cannot be in your master ROM submission.

5 Notes on Porting to HYBRID ROM

This section gives notes on porting to HYBRID ROM when using TWL-SDK.

5.1 Increase in Amount of Memory Used by the Library ---

As described in section 3.1 Increase in Amount of Memory Used by the Library, the memory consumed by the TWL-SDK libraries has increased in comparison to NITRO-SDK due to feature additions and bug fixes. HYBRID ROM applications use even more memory because of processes added to operate in TWL mode. For this reason, when an application is ported from NITRO-SDK to a HYBRID ROM application, the memory region that the application can use decreases. As an example, when past DS projects were ported to TWL-SDK 5.2, the memory consumed by the library increased by roughly 15 KB.

If a ported application has all NITRO-SDK functions linked, library memory consumption increases by about 19 KB. An increase of about 13 KB is experienced when using Thumb code. However, in cases where the number of inline functions in the library has increased, the actual amount of memory consumed may be larger than these figures because the memory used increases as the number of duplicate calls increases.

As described in section 4.4 Revised Start Address of the Static Module, with HYBRID ROM the first 16 KB of memory are reserved for the system. For this reason, the memory region that the application can use is reduced by a total of 16 KB from the beginning of the region due to the above increase in the library and for system use.

The following sections describe methods to cope with insufficient memory.

5.1.1 Disabling the Cartridge Library (CTRDG) ---

`CTRDG_Init`, which initializes the Cartridge library, is always called from `OS_Init()`, and that amount of memory is consumed. If the Cartridge library is not always used, by redefining the weak function, `CTRDG_Init`, which calls the `CTRDG_DummyInit` function internally, that memory use can be reduced by about 3 KB. For details, see **Cartridge Module (CTRDG) → Initialization → CTRDG_DummyInit** in the *Function Reference Manual*.

5.1.2 Using the Beginning Memory Region ---

Because this region, for which the first 16 KB is reserved for the system, is used only in TWL mode, the application can use this region in NITRO mode. Specifically, the region from `0x02000000` to `0x02004000` can be used only when `(OS_IsRunOnTwl() == FALSE)`.

Use the `OS_CreateExtraHeap` function to create a heap from this region and the `OS_AddExtraAreaToHeap` function to add this region to an existing heap. For details, see **Operating System (OS) → Overview → Memory Allocation: Overview** in the *Function Reference Manual*.

6 Notes on Porting to LIMITED ROM

This section gives notes on porting to LIMITED ROM when using TWL-SDK.

6.1 No Support for Clone Boots

LIMITED ROM applications do not support clone boot. Select HYBRID ROM if you want to use clone boot.

7 Notes on Porting to a NAND Application

The notes in this section are for porting to a NAND application when using TWL-SDK. NAND applications are essentially created as LIMITED ROM applications because they can run only on a TWL system. They can also be created as HYBRID ROM applications for the purpose of clone boot support.

For details on NAND application development, see the *TWL-SDK NAND Application Development Manual* (`NandAppManual.pdf`).

7.1 Linking the NA Library

To make a NAND application, you must link the NA library. Be sure to link the file `lib/ARM9-TS/*/libna.TWL*.a`.

7.2 Using OS_RebootSystem in Place of OS_ResetSystem

To use software reset with a NAND application, use the `OS_RebootSystem` function rather than the `OS_ResetSystem` function.

7.3 Return Value of OS_GetConsoleType

With NAND applications, the logical sum of `OS_CONSOLE_DEV_MASK` and the return value of the `OS_GetConsoleType` function is passed as `OS_CONSOLE_DEV_NAND`.

7.4 Provision of an Electronic Manual

Because NAND applications do not include a printed operation manual, you must include an electronic manual. This manual should be available from the title screen or soon after that. For details, see *Nintendo DS/TWL Programming Guidelines* or the TWL Manual Tools library.

7.5 About Save Data

In the case of card applications, save data is saved to a Game Card backup device. However, with NAND applications, save data is saved in system NAND memory. Notes on porting to a NAND application are given below.

7.5.1 Deleting CARD Functions

Because NAND applications do not use a Game Card backup device, be sure to delete all CARD functions.

7.5.2 Use of FS Functions

Use the FS functions on NAND application save data to access the `dataPub` and `dataPrv` archives. For details, see section 7.1 Using the Save Data Region in the *TWL-SDK NAND Application Development Manual* (`NandAppManual.pdf`).

7.5.3 User Initialization of Private Save Data

Although it is possible to copy data to an SD Card with NAND applications, private save data is not included. Private save data is therefore cleared when writing a copied NAND application back from an SD Card. Applications must therefore always anticipate private save data to be cleared.

7.5.4 About the Size Specified for Save Data

The size of save data is specified in the RSF file. (See section 7.6.2 Size Specification for Private Save Data and Public Save Data.) Because memory for managing the file system and other purposes is included in the save data region, the memory available for applications is actually smaller than the value specified. For information on the amount of memory actually available for each save data region, see **Tools** → **Tools Related to ROM Images** → **Save Data Size List** in the *Function Reference Manual*.

7.5.5 Improved Access Speed Compared to Flash Memory

The speed of reading and writing NAND memory is faster compared to the flash memory used with card applications. For details, see Chapter 23 NAND Flash Memory in the *TWL Programming Manual*.

7.5.6 Power Shutoff During Save

If the Power button is pressed during the writing of save data, the FS function aborts writing and returns `FS_RESULT_CANCELED`. The data that had been written up until that point will be accurately reflected in the save data. There is no concern about the file table being destroyed.

7.5.7 Considering the NAND Write Lifetime

Although it used to be necessary to consider the rewritable lifetime of the Game Card backup memory, this consideration is not really necessary with system NAND memory. However, be sure to avoid excessive deletions and writes (such as saving every second or saving every time a character takes a step). When performing automatic saves, be sure that the data is saved at a frequency of 128 KB or less every 3 minutes. As an alternative, you can manage data in main memory during normal operations and save it at time of shutdown using the `PM_AppendPreExitCallback` function or equivalent when power is shut off. However, even in such cases, because the shutdown process must complete within 3 seconds, be sure data can be saved quickly enough to allow shutdown within this time limit.

For details, see the *Nintendo DS/TWL Programming Guidelines*.

7.5.8 New FS Errors Requiring Handling

Unlike ROM archives, when NAND archives are accessed with FS functions the following errors must be handled.

7.5.8.1 FS_RESULT_MEDIA_FATAL

This error is issued in cases such as when a fatal device abnormality has been detected. Because continuing to process this archive would be difficult, you must prompt the user to send their unit in for repair.

7.5.8.2 FS_RESULT_BAD_FORMAT

This error is issued in cases such as when damage to the file system is detected. Because continuing to process this archive would be difficult, you must reformat this archive using the `NA_FormatTitleArchive` function.

7.6 RSF File Revisions

For NAND applications, make the settings for `TitleName`, `MakerCode`, `RemasterVersion`, `RomHeaderTemplate`, and `BannerFile` based on `$TwlSDK/include/twl/specfiles/ROM-TS_nand.rsf`.

This section covers only the points you need to be careful about. For more details, see **Tools** → **ROM image tools** → **makerom.TWL** in the *Function Reference Manual*.

7.6.1 InitialCode Setting

Set the four-character ASCII game code assigned to each application under `InitialCode` in the `AppendProperty` section of the RSF.

7.6.2 Size Specification for Private Save Data and Public Save Data

Specify the save data size to be used for private and public data in `PrivateSaveDataSize` and `PublicSaveDataSize`, respectively.

7.6.3 Revising RomSize and RomFootPadding

In NAND applications the `RomSize` specification is ignored, and instead the optimal value is selected automatically. Be sure to specify `FALSE` for `RomFootPadding`.

7.7 Notes on Using FS Functions to Access ROM Archives

Because the physical media and format differ between conventional card applications and NAND applications, there are several points to note regarding the use of FS functions to access ROM archives.

7.7.1 Addition of Random Delay to NAND Read Times

The access speed for system NAND memory is undefined due to reduced performance caused by individual hardware differences and degradation over time. A random delay time has been added to the read access time in the case of DEBUG and RELEASE builds for the purpose of virtually simulating this “undefined access speed.” Be sure to use a FINALROM build in cases where there may be obstacles to measuring performance.

7.7.2 Higher Speed Access of NAND Application ROM Archives

The read access speed for NAND application ROM archives is nearly twice as fast as a Card application using one-time PROM. The problem described in section 7.7.1 Addition of Random Delay to NAND Read Times is a factor that sometimes causes lengthy delays, but access speed will still never be as slow as the access speed of Card applications.

7.7.3 Inability to Set a DMA Channel with FS_Init

Although a DMA channel can be set for card applications using the `FS_Init` function, one cannot be set for NAND applications. This is due to the fact that use of DMA transfers is not possible for NAND.

7.7.4 Delayed RTC, SNDEX, and Wireless Processing

When accessing NAND memory using FS functions (including NA functions), RTC, SNDEX, and wireless processing is delayed until execution of the FS function completes. If a synchronous function such as `RTC_GetTime` is executed by another thread, execution is blocked until execution of the FS function finishes. To avoid this problem, use an asynchronous function such as `RTC_GetTimeAsync`. There are also plans to correct this problem in the future revision of the SDK.

7.7.5 Reduced File Access Speed Using MP Communications

When accessing a ROM archive with a card application, MP communications carried out by the ARM7 are not affected because ROM archive access is performed by the ARM9. However, MP communications are easily affected with NAND applications because NAND archives are accessed using the same ARM7 used for MP communications. File access speed is slower during MP communications compared to when MP communications are not being performed.

All company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2009 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.