Revolution Transition Guide

Version 1.10

The contents in this document are highly confidential and should be handled accordingly.

RVL-06-0025-001-B
Released: October 22, 2007

Contents

Code Examples

Revision History

Version	Date Revised	Item	Description
Version 1.10	2007/07/27	2	Deleted a description about the old development environment.
		9	Deleted a description of SRAM functions that are compatible with Nintendo GameCube.
		10	Revised a descripion about the network.
		11	Revised descriptions to match with the current SDK.
		11.1	Deleted.
Version 1.00	2006/03/22	-	First release by Nintendo of America Inc.

1 Overview

This document highlights the major differences facing developers when migrating applications from Nintendo GameCube to Revolution, with an emphasis on architectural contrasts rather than performance ones.

The Revolution platform heavily leverages Nintendo GameCube to minimize porting challenges. The OS API is nearly identical, with legacy support for most deprecated functions.

This is a living document and is subject to change as new features are enabled in the Revolution Development Environment.

2 Assumptions

This document assumes that:

- The reader is familiar with the Nintendo GameCube development environment (either GDEV or DDH).
- The reader is porting from the HW2 revision of Nintendo GameCube.

3 Memory Architecture

Nintendo GameCube features a single main memory of 24 MB (“Splash”) that is directly addressable by the CPU. Augmenting this memory is an additional 16 MB of slower ARAM, intended primarily for audio and accessible by DMA from the CPU.

The arena, heap, and memory allocation functions in the operating system manage the main memory only.

Revolution has two main memories:

- 24 MB of 1TSRAM (`MEM1`) (24 MB in development systems)
- 64 MB of GDDR3 (`MEM2`) (128 MB in development systems)

3.1 The Arenas

Both these memories are directly addressable by the CPU and are shared with other devices in the system (such as the graphics and audio cores). These memories are treated as two separate arenas. New arena APIs have been introduced to manage them.

Code 3–1 New OS Arena Functions

```
// Arena APIs for 1TSRAM/MEM1 memory.
void* OSGetMEM1ArenaHi ( void );
void* OSGetMEM1ArenaLo ( void );
void  OSSetMEM1ArenaHi ( void* newHi );
void  OSSetMEM1ArenaLo ( void* newLo );

// Arena APIs for GDDR3/MEM2 memory.
void* OSGetMEM2ArenaHi ( void );
void* OSGetMEM2ArenaLo ( void );
void  OSSetMEM2ArenaHi ( void* newHi );
void  OSSetMEM2ArenaLo ( void* newLo );
```

Note that the legacy functions are still supported. The MEM1 arena functions are simply wrappers for the original arena functions.

Code 3–2 Wrapper functions for MEM1 Arena Functions

```
#define OSGetMEM1ArenaHi(x) OSGetArenaHi(x)
#define OSGetMEM1ArenaLo(x) OSGetArenaLo(x)
#define OSSetMEM1ArenaHi(x) OSSetArenaHi(x)
#define OSSetMEM1ArenaLo(x) OSSetArenaLo(x)
```

Applications ported to Revolution will therefore operate from MEM1 by default.

3.2 Managing Heaps

While the Nintendo GameCube OS supports multiple heaps in a given arena, the heaps must be contiguous. Thus, we cannot use the legacy APIs to create or manage heaps in the MEM2 (GDDR3) arena.

The Revolution SDK introduces a new library:

(REVOLUTION_SDK_ROOT)/build/libraries/mem

This is an advanced memory manager that supports both arenas and dynamic memory allocation. Source code is provided. For more details, refer to the *Revolution Function Reference Manual* and the associated demo program.

4 Graphics Library (GX)

The graphics library is nearly identical to Nintendo GameCube.

4.1 Performance-Metric APIs

The performance-metric APIs are not final. Optimization techniques for GX are pending as well.

4.2 Graphics FIFO

Some minor API changes exist for the graphics FIFO. The following APIs have been deleted:

Code 4–1 Deleted GX FIFO APIs

```
GXGetFifoStatus()
GXSaveCPUFifo()
GXSaveGPFifo()
```

The following APIs have revised arguments and return types:

Code 4–2 Revised GX FIFO APIs

```
GXGetCPUFifo()
GXGetGPFifo()
GXGetFifoPtrs()
```

Refer to GX in the *Revolution Function Reference Manual* for more information.

4.3 Graphics FIFO Restriction

It should be noted that the FIFO is restricted to MEM1. It cannot be placed in MEM2. All other data, however, can be placed in either memory at the developer's discretion.

5 Video Interface (VI)

The VI library is nearly identical to that of Nintendo GameCube. New APIs have been added to accomodate the new trap filter and gamma correction features. Refer to VI in the *Revolution SDK Function Reference* for more information.

Note that the external frame buffer (XFB) can be placed in either `MEM1` or `MEM2`.

6 Audio Library (AX)

6.1 Frame Time and New Effects Bus

Audio frame times have been changed from 5 msec to 3 msec. This modification frees memory in the DSP core each frame, allowing the addition of an extra effects bus (`AUXC`).

This impacts the DLS conversion tool (`DLS1WT`) because its calculations for LFOs and envelopes are based on frame size. The tool has been revised. If you are using DLS samples in your application, they will need to be re-converted.

6.2 No ARAM

ARAM has been removed from the memory hierarchy and replaced by GDDR3 (`MEM2`). The associated APIs (`AR/ARQ` and `AM`) have been obsoleted and are no longer available.

The DSP can access samples from either `MEM1` or `MEM2`. Applications must be modified to perform the following functions.

- Allocate a block of memory
- Read samples into said block of memory from disc
- Notify the Sound Pipeline library (or other abstraction) of the samples' addresses

Note that the application is still responsible for specifying a zero buffer.

For examples of how to make these modifications, please refer to the AX and SP demos.

6.3 Voice Count

As of this writing, the voice count for the audio subsystem is the same as Nintendo GameCube because the DSP and AX libraries have not been optimized for the new memory hierarchy. Revised libraries are forthcoming.

7 Audio Library (AI)

The Wii console does not support hardware audio streaming from disc. The related APIs have been obsoleted.

Note that a library to support software-based audio streaming from disc is forthcoming.

8 Nintendo GameCube Memory Cards

Nintendo GameCube Memory Cards are supported through the legacy CARD library. This allows applications to import data from Nintendo GameCube titles.

Writes to Nintendo GameCube Memory Cards are permitted, but strongly discouraged for production applications. The CARD library is supplanted by the NAND API for saving and restoring game data.

For more information, please refer to the *Revolution SDK Function Reference* and the NAND demo programs.

9 System Configuration (SC)

Nintendo GameCube system preferences are accessed through a series of OS APIs. These are maintained for access to the legacy Nintendo GameCube system preferences in the SRAM.

With Revolution, you must migrate to the Revolution-specific system configuration library (SC).

The Revolution configuration parameters are a superset of the Nintendo GameCube parameters. Common parameters will be synchronized between the legacy SRAM and NAND-based system configuration file during boot.

Refer to the *Revolution SDK Function Reference* for details on the SC library. A demo program is also available.

10 Network Support and Host I/O

The Revolution SDK itself does not have network support. In order to use network features, Revolution SDK must be used in conjunction with the Revolution SDK Extension (RevoEX).

However, NDEV does support Host I/O similar to that of Nintendo GameCube. Refer to the Revolution SDK documentation for more details.

11 Controller Libraries (PAD, WPAD, and KPAD)

The Revolution platform supports legacy Nintendo GameCube Controllers via the PAD library.

To use the Wii Remote, which is the standard controller for Wii, the WPAD and KPAD libraries are used. The KPAD library is a high-level library that is positioned above the WPAD library.

For details about each of these libraries, refer to the *Revolution SDK Function Reference*.