

TWL-System Memory Allocators

An Integrated Interface for Allocating and Releasing Memory

2008/05/30

**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	5
2	Initializing Memory Allocators	6
3	Memory Allocator Functions	7
4	Associating Unique Memory Management Libraries and Memory Allocators	8

Tables

Table 2-1 Functions That Initialize the Memory Allocators	6
Table 3-1 Functions for Allocating and Releasing Memory Blocks	7

Revision History

Revision Date	Description
2008/05/30	Made revisions in line with the NITRO-System name change (from NITRO-System to TWL-System).
2008/04/08	Changed the format of the Revision History. Added support for the TWL-SDK.
2005/08/26	Revised an incorrect function name: from <code>NNS_FNDInitAllocatorForOSHeap</code> to <code>NNS_FndInitAllocatorForSDKHeap</code> .
2005/01/06	Revised the version number.
2004/08/02	Initial version.

1 Introduction

Some of the TWL-System libraries allocate necessary memory themselves. While such libraries require dynamic allocation of memory, any method of memory resource management can be used. Because it is never a good idea to depend on any one particular memory management library, the built-in memory allocation system was created and implemented.

The memory allocator is used in conjunction with a specific memory management library. The memory allocator provides only the functionality for allocating and releasing memory while the actual allocation and release operations are carried out by calling the memory management library functionality. The library receives the memory allocator and, through it, secures and releases memory. This process means that the library is not dependent on a specific memory management library to function properly.

2 Initializing Memory Allocators

Before using a memory allocator, you must initialize it. The memory allocator holds the required information in a structure named `NNSFndAllocator` and initializes by passing the pointer to this structure using the initialization function. There are four types of initialization functions: Three support the TWL-System heaps, and one supports the TWL-SDK heap. Table 2-1 describes the initialization functions.

Table 2-1 Functions That Initialize the Memory Allocators

Function	Description
<code>NNS_FndInitAllocatorForExpHeap()</code>	Initializes the allocator so that memory can be allocated and released from the extended heap.
<code>NNS_FndInitAllocatorForFrmHeap()</code>	Initializes the allocator so that memory can be allocated and released from the frame heap.
<code>NNS_FndInitAllocatorForUnitHeap()</code>	Initializes the allocator so that memory can be allocated and released from the unit heap.
<code>NNS_FndInitAllocatorForSDKHeap()</code>	Initializes the allocator so that memory can be allocated and released from the TWL-SDK heap.

Regardless of which initialization function you use, you must create the corresponding heap before calling the function.

3 Memory Allocator Functions

There are two memory allocator functions, one for allocating and the other for releasing memory. The function's actual behavior depends on the type of the memory management library associated with it. Table 3-1 describes the functions for allocating and releasing memory blocks.

Table 3-1 Functions for Allocating and Releasing Memory Blocks

Function	Description
<code>NNS_FndAllocFromAllocator()</code>	Allocates memory blocks from the allocator
<code>NNS_FndFreeToAllocator()</code>	Releases memory blocks from the allocator (returning them to the allocator)

4 Associating Unique Memory Management Libraries and Memory Allocators

Situations may arise where you want to use a unique memory management library with a memory allocator or to change the behavior of a conventionally prepared memory allocator. You can do so by uniquely implementing the memory allocator. See the memory allocator source for instructions on unique implementation.

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

© 2004-2009 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.