NAME

ddi_dma_sync – synchronize CPU and I/O views of memory

SYNOPSIS

#include <sys/conf.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

int ddi_dma_sync(ddi_dma_handle_t *handle*, **off_t** *offset*, **u_int** *length*, **u_int** *type***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

handle The *handle* filled in by a call to **ddi_dma_setup**(9F).

offset The offset into the object described by the *handle*.

length The length, in bytes, of the area to synchronize. When *length* is zero, the entire range starting from **offset** to the end of the object has the requested operation applied to it.

type Indicates the caller's desire about what view of the memory object to synchronize. The possible values are **DDI_DMA_SYNC_FORDEV**, **DDI_DMA_SYNC_FORCPU** and **DDI_DMA_SYNC_FORKERNEL**.

DESCRIPTION

ddi_dma_sync( ) is used to selectively synchronize either a DMA device's or a CPU's view of a memory object that has been mapped for I/O. This may involve operations such as flushes of CPU or I/O caches, as well as other more complex operations such as stalling until hardware write buffers have drained.

This function need only be called under certain circumstances. When a memory object is mapped for DMA , you may assume that an implicit **ddi_dma_sync( )** is done for you when you call **ddi_dma_setup( )**. When a memory object is unmapped via a call to **ddi_dma_free**(9F), you may assume that an implicit **ddi_dma_sync( )** is done for you. However, at any time between mapping a memory object for DMA and unmapping it after DMA completes, if the memory object has been modified by either the DMA device or a CPU and you wish to ensure that the change is noticed by the party that *didn't* do the modifying, a call to **ddi_dma_sync( )** is required. This is true *independent* of any attributes of the memory object including, but not limited to, whether or not the memory was allocated for non-streaming mode I/O (see **ddi_iopb_alloc**(9F)) or whether or not the memory was mapped for DMA in non-streaming mode (see **ddi_dma_req**(9S)).

This cannot be stated too strongly. If a consistent view of the memory object must be ensured between the time you map the object for DMA and the time you free such a mapping, you **must** call **ddi_dma_sync( )** to ensure that either a CPU or a DMA device has such a consistent view.

What to set *type* to depends on the view you are trying to ensure consistency for. If the memory object is modified by a CPU , and the object is going to be *read* by the DMA engine of your device, you use **DDI_DMA_SYNC_FORDEV**. This ensures that your device's DMA engine sees any changes that a CPU has made to the memory object. If the DMA engine for your device has *written* to the memory object, and you are going to *read* (with a CPU) the object (using an extant virtual address mapping that you have to the memory object), you use **DDI_DMA_SYNC_FORCPU**. This ensures that a CPU's view of the memory object includes any changes made to the object by your device's DMA engine. If you are only interested in the kernel's view (kernel-space part of the CPU's view) you may use **DDI_DMA_SYNC_FORKERNEL**. This gives a *hint* to the system—that is, if it is more economical to synchronize the kernel's view only, then do so; otherwise, synchronize for CPU.

RETURN VALUES

ddi_dma_sync( ) returns:

DDI_SUCCESS Caches are successfully flushed.

DDI_FAILURE The address range to be flushed is out of the address range established by **ddi_dma_setup**(9F).

CONTEXT

ddi_dma_sync( ) can be called from user or interrupt context.

SEE ALSO

ddi_dma_free(9F), **ddi_dma_setup**(9F), **ddi_iopb_alloc**(9F)

Writing Device Drivers