NAME

ddi_dma_mem_alloc – allocate memory for DMA transfer

SYNOPSIS

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

int ddi_dma_mem_alloc(ddi_dma_handle_t *handle*, **uint_t** *length*,
  ddi_device_acc_attr_t **accattrp*, **ulong_t** *flags*,
  int *(*waitfp)* **(caddr_t), caddr_t** *arg*, **caddr_t** **kaddrp*,
  uint_t **real_length*, **ddi_acc_handle_t** **handlep***);**

ARGUMENTS

handle — The DMA handle previously allocated by a call to **ddi_dma_alloc_handle**(9F).

length — The length in bytes of the desired allocation.

accattrp — Pointer to a device access attribute structure of this device (see **ddi_device_acc_attr**(9S)).

flags — Data transfer mode flags. Possible values are:

- **DDI_DMA_STREAMING** — Sequential, unidirectional, block-sized, and block-aligned transfers.
- **DDI_DMA_CONSISTENT** — Nonsequential transfers of small objects.

waitfp — The address of a function to call back later if resources are not available now. The special function addresses **DDI_DMA_SLEEP** and **DDI_DMA_DONTWAIT** are taken to mean, respectively, wait until resources are available or, do not wait and do not schedule a callback.

arg — Argument to be passed to the callback function, if such a function is specified.

kaddrp — On successful return, **kaddrp* points to the allocated memory.

real_length* — The amount of memory, in bytes, allocated. Alignment and padding requirements may require **ddi_dma_mem_alloc( ) to allocate more memory than requested in *length*.

handlep — Pointer to a data access handle.

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

DESCRIPTION

ddi_dma_mem_alloc( ) allocates memory for DMA transfers to or from a device. The allocation will obey the alignment, padding constraints and device granularity as specified by the DMA attributes (see **ddi_dma_attr**(9S)) passed to **ddi_dma_alloc_handle**(9F) and the more restrictive attributes imposed by the system.

flags should be set to **DDI_DMA_STREAMING** if the device is doing sequential, unidirectional, block-sized, and block-aligned transfers to or from memory. The alignment and padding constraints specified by the *minxfer* and *burstsizes* fields in the DMA attribute structure, **ddi_dma_attr**(9S) (see **ddi_dma_alloc_handle**(9F)) will be used to allocate the most effective hardware support for large transfers. For example, if an I/O transfer can be sped up by using an I/O cache, which has a minimum transfer of one cache line, **ddi_dma_mem_alloc( )** will align the memory at a cache line boundary and it will round up **real_length* to a multiple of the cache line size.

flags should be set to **DDI_DMA_CONSISTENT** if the device accesses memory randomly, or if synchronization steps using **ddi_dma_sync**(9F) need to be as efficient as possible. I/O parameter blocks used for communication between a device and a driver should be allocated using **DDI_DMA_CONSISTENT**.

The device access attributes are specified in the location pointed by the *accattrp* argument (see **ddi_device_acc_attr**(9S)).

The data access handle is returned in *handlep*. *handlep* is opaque – drivers may not attempt to interpret its value. To access the data content, the driver must invoke **ddi_getb**(9F) or **ddi_getb**(9F) (depending on the data transfer direction) with the data access handle.

DMA resources must be established before performing a DMA transfer by passing *kaddrp* and **real_length* as returned from **ddi_dma_mem_alloc( )** and the flag **DDI_DMA_STREAMING** or **DDI_DMA_CONSISTENT** to **ddi_dma_addr_bind_handle**(9F). In addition, to ensure the consistency of a memory object shared between the CPU and the device after a DMA transfer, explicit synchronization steps using **ddi_dma_sync**(9F) or **ddi_dma_unbind_handle**(9F) are required.

RETURN VALUES

ddi_dma_mem_alloc( ) returns:

DDI_SUCCESS	Memory successfully allocated.
DDI_FAILURE	Memory allocation failed.

CONTEXT

ddi_dma_mem_alloc( ) can be called from user or interrupt context, except when *waitfp* is set to **DDI_DMA_SLEEP**, in which case it can be called from user context only.

SEE ALSO

ddi_dma_addr_bind_handle(9F), **ddi_dma_alloc_handle**(9F), **ddi_dma_unbind_handle**(9F), **ddi_dma_mem_free**(9F), **ddi_dma_sync**(9F), **ddi_getb**(9F), **ddi_putb**(9F)

Writing Device Drivers