NAME

ddi_dma_setup – setup DMA resources

SYNOPSIS

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

int ddi_dma_setup(dev_info_t **dip*, **ddi_dma_req_t** **dmareqp*,
    ddi_dma_handle_t **handlep***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

dip — A pointer to the device's **dev_info** structure.

dmareqp — A pointer to a DMA request structure (see **ddi_dma_req**(9S)).

handlep — A pointer to a DMA handle to be filled in. See below for a discussion of a handle. If *handlep* is **NULL**, the call to **ddi_dma_setup( )** is considered an advisory call, in which case no resources are allocated, but a value indicating the legality and the feasibility of the request is returned.

DESCRIPTION

ddi_dma_setup( ) allocates resources for a memory object such that a device can perform DMA to or from that object.

A call to **ddi_dma_setup( )** informs the system that device referred to by *dip* wishes to perform DMA to or from a memory object. The memory object, the device's DMA capabilities, the device driver's policy on whether to wait for resources, are all specified in the **ddi_dma_req** structure pointed to by *dmareqp*.

A successful call to **ddi_dma_setup( )** fills in the value pointed to by *handlep*. This is an opaque object called a DMA handle. This handle is then used in subsequent DMA calls, until **ddi_dma_free**(9F) is called.

Again a DMA handle is opaque—drivers may **not** attempt to interpret its value. When a driver wants to enable its DMA engine, it must retrieve the appropriate address to supply to its DMA engine using a call to **ddi_dma_htoc**(9F), which takes a pointer to a DMA handle and returns the appropriate DMA address.

When DMA transfer completes, the driver should free up the the allocated DMA resources by calling **ddi_dma_free( )**.

RETURN VALUES

ddi_dma_setup( ) returns:

DDI_DMA_MAPPED — Successfully allocated resources for the object. In the case of an *advisory* call, this indicates that the request is legal.

DDI_DMA_PARTIAL_MAP — Successfully allocated resources for a *part* of the object. This is acceptable when partial transfers are allowed using a flag setting in the **ddi_dma_req** structure (see **ddi_dma_req**(9S) and **ddi_dma_movwin**(9F)).

DDI_DMA_NORESOURCES — When no resources are available.

DDI_DMA_NOMAPPING	The object cannot be reached by the device requesting the resources.
DDI_DMA_TOOBIG	The object is too big and exceeds the available resources. The maximum size varies depending on machine and configuration.

CONTEXT

ddi_dma_setup( ) can be called from user or interrupt context, except when the **dmar_fp** member of the **ddi_dma_req** structure pointed to by *dmareqp* is set to **DDI_DMA_SLEEP,** in which case it can be called from user context only.

SEE ALSO

ddi_dma_addr_setup(9F), **ddi_dma_buf_setup**(9F), **ddi_dma_free**(9F), **ddi_dma_htoc**(9F), **ddi_dma_sync**(9F), **ddi_dma_req**(9S)

Writing Device Drivers

NOTES

The construction of the **ddi_dma_req** structure is complicated. Use of the provided interface functions such as **ddi_dma_buf_setup**(9F) simplifies this task.