NAME

ddi_dma_lim_sparc, ddi_dma_lim – SPARC DMA limits structure

SYNOPSIS

```
#include <sys/ddidmareq.h>
```

INTERFACE LEVEL

Solaris SPARC DDI specific (Solaris SPARC DDI).

DESCRIPTION

A **ddi_dma_lim** structure describes in a generic fashion the possible limitations of a device's DMA engine. This information is used by the system when it attempts to set up DMA resources for a device.

STRUCTURE MEMBERS

```
u_long   dlim_addr_lo;     /* low range of 32 bit addressing capability */
u_long   dlim_addr_hi;     /* inclusive upper bound of addressing */
                           /* capability */
u_int    dlim_cntr_max;    /* inclusive upper bound of dma engine's */
                           /* address limit * /
u_int    dlim_burstsizes;  /* binary encoded dma burst sizes */
u_int    dlim_minxfer;     /* minimum effective dma transfer size */
u_int    dlim_dmaspeed;    /* average dma data rate (kb/s) */
```

The **dlim_addr_lo** and **dlim_addr_hi** fields specify the address range the device's DMA engine can access. The **dlim_addr_lo** field describes the lower 32 bit boundary of the device's DMA engine, the **dlim_addr_hi** describes the inclusive upper 32 bit boundary. The system will allocate DMA resources in a way that the address for programming the device's DMA engine (see **ddi_dma_cookie**(9S) or **ddi_dma_htoc**(9F)) will be within this range. For example, if your device can access the whole 32 bit address range, you may use [**0**,**0xFFFFFFFF**]. If your device has just a 16 bit address register but will access the top of the 32 bit address range, then [**0xFFFF0000**,**0xFFFFFFFF**] would be the right limit.

The **dlim_cntr_max** field describes an inclusive upper bound for the device's DMA engine address register. This handles a fairly common case where a portion of the address register is simply a latch rather than a full register. For example, the upper 8 bits of a 32 bit address register may be a latch. This splits the address register into a portion which acts as a true address register (24 bits) for a 16 megabyte segment and a latch (8 bits) to hold a segment number. To describe these limits, you would specify **0xFFFFFF** in the **dlim_cntr_max** structure.

The **dlim_burstsizes** field describes the possible burst sizes the device's DMA engine can accept. At the time of a DMA resource request, this element defines the possible DMA burst cycle sizes that the requester's DMA engine can handle. The format of the data is binary encoding of burst sizes assumed to be powers of two. That is, if a DMA engine is capable of doing 1, 2, 4 and 16 byte transfers, the encoding would be 0x17. If the device is an SBus device and can take advantage of a 64 bit SBus, the lower 16 bits are used to specify the burst size for 32 bit transfers and the upper 16 bits are used to specify the burst size for 64 bit transfers. As the resource request is handled by the system, the burstsizes value may be modified. Prior to enabling DMA for the specific device, the driver that owns the DMA engine should check (using **ddi_dma_burstsizes**(9F)) what the allowed

burstsizes have become and program the DMA engine appropriately.

The **dlim_minxfer** field describes the minimum effective DMA transfer size (in units of bytes). It must be a power of two. This value specifies the minimum effective granularity of the DMA engine. It is distinct from **dlim_burstsizes** in that it describes the minimum amount of access a DMA transfer will effect. **dlim_burstsizes** describes in what electrical fashion the DMA engine might perform its accesses, while **dlim_minxfer** describes the minimum amount of memory that can be touched by the DMA transfer. As a resource request is handled by the system, the **dlim_minxfer** value may be modified contingent upon the presence (and use) of I/O caches and DMA write buffers in between the DMA engine and the object that DMA is being performed on. After DMA resources have been allocated, the resultant minimum transfer value can be gotten using **ddi_dma_devalign**(9F).

The field **dlim_dmaspeed** is the expected average data rate for the DMA engine (in units of kilobytes per second). Note that this should not be the maximum, or peak, burst data rate, but a reasonable guess as to the average throughput. This field is entirely optional, and may be left as zero. Its intended use is to provide some hints about how much DMA resources this device may need.

SEE ALSO **ddi_dma_addr_setup**(9F), **ddi_dma_buf_setup**(9F), **ddi_dma_burstsizes**(9F), **ddi_dma_devalign**(9F), **ddi_dma_lim_x86**(9S), **ddi_dma_req**(9S), **ddi_dma_setup**(9F)