NAME

ddi_dma_attr – DMA attributes structure

SYNOPSIS

#include <sys/ddidmareq.h>

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

DESCRIPTION

A **ddi_dma_attr_t** structure describes device and DMA engine specific attributes necessary to allocate DMA resources for a device. The driver may have to extend the attributes with bus specific information depending on the bus to which the device is connected.

STRUCTURE MEMBERS

```
uint_t               dma_attr_version;
unsigned long long   dma_attr_addr_lo;
unsigned long long   dma_attr_addr_hi;
unsigned long long   dma_attr_count_max;
unsigned long long   dma_attr_align;
uint_t               dma_attr_burstsizes;
uint_t               dma_attr_minxfer;
unsigned long long   dma_attr_maxxfer;
unsigned long long   dma_attr_seg;
int                  dma_attr_sgllen;
uint_t               dma_attr_granular;
int                  dma_attr_flags;
```

dma_attr_version stores the version number of this DMA attribute structure. It should be set to **DMA_ATTR_V0**.

The **dma_attr_addr_lo** and **dma_attr_addr_hi** fields specify the address range the device's DMA engine can access. The **dma_attr_addr_lo** field describes the inclusive lower 64 bit boundary. The **dma_attr_addr_hi** describes the inclusive upper 64 bit boundary. The system will ensure that allocated DMA resources are within the range specified. (see **ddi_dma_cookie**(9S), **ddi_dma_nextcookie**(9F))

The **dma_attr_count_max** describes an inclusive upper bound for the device's DMA counter register. For example, 0xFFFFFF would describe a DMA engine with a 24 bit counter register. DMA resource allocation functions have to break up a DMA object into multiple DMA cookies if the size of the object exceeds the size of the DMA counter register.

The **dma_attr_align** specifies alignment requirements for allocated DMA resources. This field can be used to force more restrictive alignment than specified by **dma_attr_burstsizes** or **dma_attr_minxfer**, such as alignment at a page boundary.

The **dma_attr_burstsizes** field describes the possible burst sizes the device's DMA engine can accept. The format of the data sizes is binary encoded in terms of powers of two. When DMA resources are allocated, the system may modify the burstsizes value to reflect the system limits. The driver must use the allowable burstsizes to program the DMA engine (see **ddi_dma_burstsizes**(9F)).

The **dma_attr_minxfer** field describes the minimum effective DMA access size in units of bytes. DMA resources may be modified depending on the presence and use of I/O caches and write buffers between the DMA engine and the memory object. This field is used to determine alignment and padding requirements for **ddi_dma_mem_alloc**(9F).

The **dma_attr_maxxfer** field describes the maximum effective DMA access size in units of bytes.

The **dma_attr_seg** field specifies segment boundary restrictions for allocated DMA resources. The system will allocate DMA resources for the device such that the object does not span the segment boundary specified by **dma_attr_seg**. For example a value of **0xFFFF** means DMA resources must not cross a 64K boundary. DMA resource allocation functions may have to break up a DMA object into multiple DMA cookies to enforce segment boundary restrictions. In this case, the transfer must be performed using scatter-gather I/O or multiple DMA windows.

The **dma_attr_sgllen** field describes the length of the device's DMA scatter/gather list. Possible values are as follows:

< 0 Device DMA engine is not constrained by the size – for example, DMA chaining.

= 0 Reserved.

= 1 Device DMA engine does not support scatter/gather such as third party DMA, etc.

> 1 Device DMA engine uses scatter/gather. **dma_attr_sgllen** is the maximum number of entries in the list.

The **dma_attr_granular** field describes the granularity of the device transfer size, in units of bytes. When the system allocates DMA resources, a single segment's size will be a multiple of the device granularity. Or if **dma_attr_sgllen** is larger than 1 within a window, the sum of the sizes for a subgroup of segments will be a multiple of the device granularity.

The **dma_attr_flags** field is reserved for future use. It must be set to 0.

EXAMPLES

For example, assume a device has the following DMA characteristics:

- Full 32-bit range addressable
- 24-bit address register
- 4-byte word alignment
- 4 and 8-byte burst sizes support
- Minimum effective transfer size of 1 bytes
- 64M maximum transfer size limit
- Maximum segment size of 32K
- 17 scatter/gather list elements
- 512 byte device transfer size granularity

The corresponding **ddi_dma_attr_t** structure would be initialized as follows:

```
static ddi_dma_attr_t dma_attrs = {
        DMA_ATTR_V0,                        /* version number */
        (unsigned long long)0x0,            /* low address */
        (unsigned long long)0xffffffff,     /* high address */
        (unsigned long long)0xffffff,       /* address counter max */
        (unsigned long long)0x04            /* alignment */
        0x0c,                               /* burst sizes */
        0x1,                                /* minimum transfer size */
        (unsigned long long)0x3ffffff,      /* maximum transfer size */
        (unsigned long long)0x7fff,         /* maximum segment size  */
        17,                                 /* scatter/gather list lgth */
        512,                                /* granularity */
        0                                   /* flags */
};
```

SEE ALSO

ddi_dma_burstsizes(9F), **ddi_dma_nextcookie**(9F), **ddi_dma_cookie**(9S)

Writing Device Drivers