NAME

ddi_add_intr, ddi_get_iblock_cookie, ddi_remove_intr – hardware interrupt handling routines

SYNOPSIS

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

int ddi_get_iblock_cookie(dev_info_t **dip*, **u_int** *inumber*,
ddi_iblock_cookie_t **iblock_cookiep***);**

int ddi_add_intr(dev_info_t **dip*, **u_int** *inumber*, **ddi_iblock_cookie_t** **iblock_cookiep*,
ddi_idevice_cookie_t **idevice_cookiep*, **u_int (****int_handler***)(caddr_t),**
caddr_t *int_handler_arg***);**

void ddi_remove_intr(dev_info_t **dip*, **u_int** *inumber*,
ddi_iblock_cookie_t *iblock_cookie***);**

INTERFACE LEVEL

Solaris DDI specific (Solaris DDI).

ARGUMENTS

ddi_get_iblock_cookie( )

dip	Pointer to **dev_info** structure.
inumber	Interrupt number.
iblock_cookiep	Pointer to an interrupt block cookie.

ddi_add_intr( )

dip	Pointer to **dev_info** structure.
inumber	Interrupt number.
iblock_cookiep	Optional pointer to an interrupt block cookie where a returned interrupt block cookie is stored.
idevice_cookiep	Optional pointer to an interrupt device cookie where a returned interrupt device cookie is stored.
int_handler	Pointer to interrupt handler.
int_handler_arg	Argument for interrupt handler.

ddi_remove_intr( )

dip	Pointer to **dev_info** structure.
inumber	Interrupt number.
iblock_cookie	Block cookie which identifies the interrupt handler to be removed.

DESCRIPTION

ddi_get_iblock_cookie( )

ddi_get_iblock_cookie( ) retrieves the interrupt block cookie associated with a particular interrupt specification. This routine should be called before **ddi_add_intr( )** to retrieve the interrupt block cookie needed to initialize locks (**mutex**(9F), **rwlock**(9F)) used by the interrupt routine. The interrupt number *inumber* determines which interrupt specification to retrieve the cookie for. *inumber* is associated with information provided either by the device (see **sbus**(4)) or the hardware configuration file (see **vme**(4),

sysbus(4), **isa**(4), **eisa**(4), **mca**(4), and **driver.conf**(4)). If only one interrupt is associated with the device, *inumber* should be **0**.

On a successful return, ∗*iblock_cookiep* contains information needed for initializing locks associated with the interrupt specification corresponding to *inumber* (see **mutex_init**(9F) and **rw_init**(9F)). The driver can then initialize locks acquired by the interrupt routine before calling **ddi_add_intr( )** which prevents a possible race condition where the driver's interrupt handler is called immediately *after* the driver has called **ddi_add_intr( )** but *before* the driver has initialized the locks. This may happen when an interrupt for a different device occurs on the same interrupt level. If the interrupt routine acquires the lock before the lock has been initialized, undefined behavior may result.

ddi_add_intr( )

ddi_add_intr( ) adds an interrupt handler to the system. The interrupt number *inumber* determines which interrupt the handler will be associated with. (Refer to **ddi_get_iblock_cookie( )** above.)

On a successful return, *iblock_cookiep* contains information used for initializing locks associated with this interrupt specification (see **mutex_init**(9F) and **rw_init**(9F)). Note that the interrupt block cookie is usually obtained using **ddi_get_iblock_cookie( )** to avoid the race conditions described above (refer to **ddi_get_iblock_cookie( )** above). For this reason, *iblock_cookiep* is no longer useful and should be set to **NULL**.

On a successful return, *idevice_cookiep* contains a pointer to a **ddi_idevice_cookie_t** structure (see **ddi_idevice_cookie**(9S)) containing information useful for some devices that have programmable interrupts. If *idevice_cookiep* is set to **NULL**, no value is returned.

The routine *intr_handler*, with its argument *int_handler_arg*, is called upon receipt of the appropriate interrupt. The interrupt handler should return **DDI_INTR_CLAIMED** if the interrupt was claimed, **DDI_INTR_UNCLAIMED** otherwise.

If successful, **ddi_add_intr( )** will return **DDI_SUCCESS;** if the interrupt information cannot be found, it will return **DDI_INTR_NOTFOUND.**

ddi_remove_intr( )

ddi_remove_intr( ) removes an interrupt handler from the system. Unloadable drivers should call this routine during their **detach**(9E) routine to remove their interrupt handler from the system.

The device interrupt routine for this instance of the device will not execute after **ddi_remove_intr( )** returns. **ddi_remove_intr( )** may need to wait for the device interrupt routine to complete before returning. Therefore, locks acquired by the interrupt handler should not be held across the call to **ddi_remove_intr( )** or deadlock may result.

RETURN VALUES

ddi_add_intr( ) and **ddi_get_iblock_cookie( )** return:

DDI_SUCCESS	on success.
DDI_INTR_NOTFOUND	on failure to find the interrupt.

CONTEXT

ddi_add_intr( ), **ddi_remove_intr( )**, and **ddi_get_iblock_cookie( )** can be called from user or kernel context.

SEE ALSO

driver.conf(4), **eisa**(4), **isa**(4), **mca**(4), **sbus**(4), **sysbus**(4), **vme**(4), **attach**(9E), **detach**(9E), **ddi_intr_hilevel**(9F), **mutex_init**(9F), **rw_init**(9F), **ddi_idevice_cookie**(9S)

Writing Device Drivers

BUGS

The *idevice_cookiep* should really point to a data structure that is specific to the bus architecture that the device operates on. Currently only VMEbus and SBus are supported and a single data structure is used to describe both.