

**NAME**

**iflibdd** - Device Dependent Configuration Functions

**SYNOPSIS**

```
#include <ifdi_if.h>
```

**Soft Queue Setup and Teardown Functions****Mandatory Functions**

*int*

```
ifdi_tx_queues_alloc(if_ctx_t ctx, caddr_t *vaddrs, uint64_t *paddrs, int ntxqs, int ntxqsets);
```

*int*

```
ifdi_rx_queues_alloc(if_ctx_t ctx, caddr_t *vaddrs, uint64_t *paddrs, int nrxqs, int nrxqsets);
```

*int*

```
ifdi_queues_free(if_ctx_t ctx);
```

**Optional Functions**

*int*

```
ifdi_txq_setup(if_ctx_t ctx, uint16_t qid);
```

*int*

```
ifdi_rxq_setup(if_ctx_t ctx, uint16_t qid);
```

**Device Setup and Teardown Functions****Mandatory Functions**

*int*

```
ifdi_attach_pre(if_ctx_t ctx);
```

*int*

```
ifdi_attach_post(if_ctx_t ctx);
```

*int*

```
ifdi_detach(if_ctx_t ctx);
```

**Optional Functions**

*void*

```
ifdi_vlan_register(if_ctx_t ctx, uint16_t vtag);
```

*void*  
**ifdi\_vlan\_unregister**(*if\_ctx\_t ctx, uint16\_t vtag*);

*int*  
**ifdi\_suspend**(*if\_ctx\_t ctx*);

*int*  
**ifdi\_resume**(*if\_ctx\_t ctx*);

## Device Configuration Functions

### Mandatory Functions

*void*  
**ifdi\_init**(*if\_ctx\_t ctx*);

*void*  
**ifdi\_stop**(*if\_ctx\_t ctx*);

*void*  
**ifdi\_multi\_set**(*if\_ctx\_t ctx*);

*int*  
**ifdi\_mtu\_set**(*if\_ctx\_t ctx, uint32\_t mtu*);

*void*  
**ifdi\_media\_status**(*if\_ctx\_t ctx, struct ifmediareq \*ifr*);

*int*  
**ifdi\_media\_change**(*if\_ctx\_t ctx*);

*void*  
**ifdi\_promisc\_set**(*if\_ctx\_t ctx, int flags*);

*uint64\_t*  
**ifdi\_get\_counter**(*if\_ctx\_t ctx, ift\_counter cnt*);

*void*  
**ifdi\_update\_admin\_status**(*if\_ctx\_t ctx*);

### Optional Functions

*void*  
**ifdi\_media\_set**(*if\_ctx\_t ctx*);

### Interrupt enable/disable

### Mandatory Functions

*void*  
**ifdi\_intr\_enable**(*if\_ctx\_t ctx*);

*void*  
**ifdi\_queue\_intr\_enable**(*if\_ctx\_t ctx, uint16\_t qid*);

*void*  
**ifdi\_intr\_disable**(*if\_ctx\_t ctx*);

### IOV Support

*init*  
**iovm\_init**(*if\_ctx\_t ctx, uint16\_t num\_vfs, const nvlist\_t \*params*);

*void*  
**iovm\_uinit**(*if\_ctx\_t ctx*);

*void*  
**ifdi\_vflr\_handle**(*if\_ctx\_t ctx*);

*int*  
**ifdi\_vf\_add**(*if\_ctx\_t ctx, uint16\_t vfnum, const nvlist\_t \*params*);

### Optional Functions

*void*  
**ifdi\_link\_intr\_enable**(*if\_ctx\_t ctx*);

### Optional Service Routines

*void*  
**ifdi\_timer**(*if\_ctx\_t ctx*);

*void*  
**ifdi\_watchdog\_reset**(*if\_ctx\_t ctx*);

### Additional Functions

*void*

**ifdi\_led\_func**(*if\_ctx\_t ctx, int onoff*);

*int*

**ifdi\_sysctl\_int\_delay**(*if\_ctx\_t ctx, if\_int\_delay\_info\_t iidi*);

*int*

**ifdi\_i2c\_req**(*if\_ctx\_t ctx, struct ifi2creq \*req*);

## FUNCTIONS

The above named functions are device dependent configuration functions. These routines are registered with iflib by the driver and are called from the corresponding iflib function to configure device specific functions and registers.

### Device Dependent Functions

#### Soft Queue Setup and Teardown

##### **ifdi\_tx\_queues\_alloc()**

Mandatory function that is called during iflib\_attach to allocate transmit queues. vaddrs and paddrs are arrays of virtual and physical addresses respectively of the hardware transmit queues. ntxqs is the number of queues per qset. ntxqsets is the number of qsets.

##### **ifdi\_rx\_queues\_alloc()**

Mandatory function that is called during iflib\_attach to allocate receive queues. vaddrs and paddrs are arrays of virtual and physical addresses respectively of the hardware receive queues. nrxqs is the number of queues per qset. nrxqsets is the number of qsets.

##### **ifdi\_queues\_free()**

Mandatory function that frees the allocated queues and associated transmit buffers.

##### **ifdi\_txq\_setup()**

Optional function for each transmit queue that handles device specific initialization.

##### **ifdi\_rxq\_setup()**

Optional function for each receive queue that handles device specific initialization.

### Device Setup and Teardown

##### **ifdi\_attach\_pre()**

Mandatory function implemented by the driver to perform any attach logic that precedes interrupt and queue allocation, queue setup, and interrupt assignment.

**ifdi\_attach\_post()**

Mandatory function implemented by the driver to perform any attach logic that occurs after `ifdi_attach_pre`, and `iflib`'s queue setup and MSI/MSIX(X) or legacy interrupt assignment.

**ifdi\_detach()**

Mandatory function that frees any resources allocated by the driver in `ifdi_attach_pre` and `ifdi_attach_post`.

**ifdi\_vlan\_register()**

Optional function called by the VLAN config eventhandler. *vtag* is the new VLAN tag.

**ifdi\_vlan\_unregister()**

Optional function called by the VLAN unconfig eventhandler.

**ifdi\_suspend()**

Optional function that suspends the driver.

**ifdi\_resume()**

Optional function that resumes a driver.

**Device Configuration Functions****ifdi\_init()**

Mandatory function that will initialize and bring up the hardware. For example, it will reset the chip and enable the receiver unit. It should mark the interface running, but not active ( `IFF_DRV_RUNNING`, `~IFF_DRV_OACTIVE` ).

**ifdi\_stop()**

Mandatory function that should disable all traffic on the interface by issuing a global reset on the MAC and deallocating the TX and RX buffers.

**ifdi\_multi\_set()**

Programs the interfaces multicast addresses

**ifdi\_media\_status()**

Media Ioctl Callback. Function is called whenever the user queries the status of the interface using `ifconfig(8)`. The driver sets the appropriate link type and speed in `ifmr->ifm_active`.

**ifdi\_mtu\_set()**

Sets the mtu interface to the value of the second function parameter `mtu`.

**ifdi\_media\_change()**

Function is called when the user changes speed/duplex using the media/mediaopt option with ifconfig(8).

**ifdi\_promisc\_set()**

Enables or disables promisc settings depending upon the flags value. *flags* contains the interface's ifnet(9) flags.

**ifdi\_get\_counter()**

Returns the value for counter cnt depending upon counter type.

**ifdi\_update\_admin\_status()**

Sets the link\_up state to TRUE or FALSE depending upon the OS link state. A real check of the hardware only happens with a link interrupt.

**ifdi\_media\_set()**

Need to define

**Interrupt Enable/Disable****ifdi\_intr\_enable()**

Mandatory function that enables all interrupts.

**ifdi\_intr\_disable()**

Mandatory function that disables all interrupts.

**ifdi\_queue\_intr\_enable()**

Mandatory function that enables interrupts on queue qid.

**iov\_init()**

Initialize num\_vfs VFs.

**io\_uninit()**

Tear down the context for all VFs.

**ifdi\_vflr\_handle()**

Handle any VFs that have reset themselves via a Function Level Reset (FLR).

**ifdi\_vf\_add()**

Set parameters in params in VF vnum.

**Service Routines****ifdi\_timer()**

Optional timer routine that will be run every 500ms.

**ifdi\_watchdog\_reset()**

Optional function to run when a transmit queue is hung.

**Additional Functions****ifdi\_led\_func()****ifdi\_sysctl\_int\_delay()****ifdi\_i2c\_req()****SEE ALSO**

ifconfig(8), iflibdi(9), iflibtxrx(9), ifnet(9)

**AUTHORS**

This manual page was written by Nicole Graziano