

NAME

iflibtxrx - Device Dependent Transmit and Receive Functions

SYNOPSIS

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

Interface Manipulation Functions

int

```
isc_txd_encap(void *sc, if_pkt_info_t pi);
```

void

```
isc_txd_flush(void *sc, uint16_t qid, uint32_t _pid_x_or_credits_);
```

int

```
isc_txd_credits_update(void *sc, uint16_t qid, bool clear);
```

int

```
isc_rxd_available(void *sc, uint16_t qsid, uint32_t cid_x);
```

void

```
isc_rxd_refill(void *sc, uint16_t qsid, uint8_t flid, uint32_t pid_x, uint64_t *paddrs, caddr_t *vaddrs,
               uint16_t count);
```

void

```
isc_rxd_flush(void *sc, uint16_t qsid, uint8_t flid, uint32_t pid_x);
```

int

```
isc_rxd_pkt_get(void *sc, if_rxd_info_t ri);
```

Global Variables

extern struct if_txrx

DATA STRUCTURES

The device dependent mechanisms for handling packet transmit and receive are primarily defined by the functions named above. The `if_pkt_info` data structure contains statistics and identifying info necessary for packet transmission. While the data structure for packet receipt is the `if_rxd_info` structure.

The `if_pkt_info` Structure

The fields of `struct if_pkt_info` are as follows:

- ipi_len* (*uint32_t*) Denotes the size of packet to be sent on the transmit queue.
- ipi_segs* (*bus_dma_segment_t* *) A pointer to the *bus_dma_segment* of the device independent transfer queue defined in *iflib*.
- ipi_qsidx* Unique index value assigned sequentially to each transmit queue. Used to reference the currently transmitting queue.
- ipi_nsegs* (*uint16_t*) Number of descriptors to be read into the device dependent transfer descriptors.
- ipi_ndescs* (*uint16_t*) Number of descriptors in use. Calculated by subtracting the old *pidx* value from the new *pidx* value.
- ipi_flags* (*uint16_t*) Flags defined on a per packet basis.
- ipi_pidx* (*uint32_t*) Value of first *pidx* sent to the *isc_encap* function for encapsulation and subsequent transmission.
- ipi_new_pidx* (*uint32_t*) Value set after the termination of the *isc_encap* function. This value will become the first *pidx* sent to the *isc-encap* the next time that the function is called.

The Following Fields Are Used For Offload Handling

- ipi_csum_flags* (*uint64_t*) Flags describing the checksum values, used on a per packet basis.
- ipi_tso_segsz* (*uint16_t*) Size of the TSO Segment Size.
- ipi_mflags* (*uint16_t*) Flags describing the operational parameters of the mbuf.
- ipi_vtag* (*uint16_t*) Contains the VLAN information in the Ethernet Frame.
- ipi_etype* (*uint16_t*) Type of ethernet header protocol as contained by the struct *ether_vlan_header*.
- ipi_ehrdlen* (*uint8_t*) Length of the Ethernet Header.
- ipi_ip_hlen* (*uint8_t*) Length of the TCP Header

ipi_tcp_hlen (*uint8_t*) Length of the TCP Header.

ipi_tcp_hflags

(*uint8_t*) Flags describing the operational parameters of the TCP Header.

ipi_ipproto (*uint8_t*) Specifies the type of IP Protocol in use. Example TCP, UDP, or SCTP.

The *if_rxd_info* Structure

The fields of *struct if_rxd_info* are as follows:

iri_qsidx (*uint16_t*) Unique index value assigned sequentially to each receive queue. Used to reference the currently receiving queue.

iri_vtag (*uint16_t*) Contains the VLAN information in the Ethernet Frame.

iri_len (*uint16_t*) Denotes the size of a received packet.

iri_next_offset

(*uint16_t*) Denotes the offset value for the next packet to be receive. A Null value signifies the end of packet.

iri_cidx (*uint32_t*) Denotes the index value of the packet currently being processed in the consumer queue.

iri_flowid (*uint32_t*) Value of the RSS hash for the packet.

iri_flags (*uint*)

Flags describing the operational parameters of the mbuf contained in the receive packet.

iri_csum_flags

(*uint32_t*) Flags describing the checksum value contained in the receive packet.

iri_csum_data (*uint32_t*) Checksum data contained in the mbuf(9) packet header.

iri_m (*struct mbuf **) A mbuf for drivers that manage their own receive queues.

iri_ifp (*struct ifnet **) A link back to the interface structure. Utilized by drivers that have multiple interface per softc.

- iri_rstype* (*uint8_t*) The value of the RSS hash type.
- iri_pad* (*uint8_t*) The length of any padding contained by the received data.
- iri_qidx* (*uint8_t*) Represents the type of queue event. If value ≥ 0 then it is the freelist id otherwise it is a completion queue event.

FUNCTIONS

All function calls are associated exclusively with either packet transmission or receipt. The void **sc* passed as the first argument to all of the following functions represents the driver's softc.

Transmit Packet Functions

isc_txd_encap()

Transmit function that sends a packet on an interface. The *if_pkt_info* data structure contains data information fields describing the packet. This function returns 0 if successful, otherwise an error value is returned.

isc_txd_flush()

Flush function is called immediately after the *isc_txd_encap* function transmits a packet. It updates the hardware producer index or increments the descriptors used to *pidx_or_credits* in the queue designated by the *qid* number. This is often referred to as poking the doorbell register.

isc_txd_credits_update()

Credit function advances the buffer ring and calculates the credits (descriptors) processed. Until the I/O is complete it cleans the range in case of multisegments and updates the count of processed packets. The function returns the number of processed credits.

Receive Packet Functions

isc_rxd_available()

Function calculates the remaining number of descriptors from a position given by *idx*. The function returns this value.

isc_rxd_refill()

Starting with the physical address *paddrs*, the function reads a packet into the *rx_ring* until a value designated by *count* is reached. *vaddrs* is typically not needed and is provided for devices that place their own metadata in the packet header.

isc_rxd_flush()

Flush function updates the producer pointer on the free list *flid* in queue set number *qid* to *pidx* to reflect the presence of new buffers.

isc_rxd_pkt_get()

Process a single software descriptor. `rxr->rx_base[i]` contains a descriptor that describes a received packet. Hardware specific information about the buffer referred to by `ri` is returned in the data structure `if_rxd_info`

SEE ALSO

`iflibdd(9)`, `iflibdi(9)`, `mbuf(9)`

AUTHORS

This manual page was written by Nicole Graziano