allocateNetbuf

finishInitialization
outputPacket:address:
performCommand:data:

free
initFromDeviceDescription:
attachToNetworkWithAddress:

Handling interrupts interruptOccurred (IODirectDevice)

Transmitting packets transmit:

performLoopback:

Setting and handling hardware timeouts

setRelativeTimeout:
relativeTimeout
clearTimeout
timeoutOccurred (IODirectDevice)

Setting and getting the state of the hardware

isRunning

(void)addMulticastAddress:(enet_addr_t *)address

Does nothing. Subclasses that support multicast mode can implement this method so that it notifies should receive packets sent to address. Some subclasses that support multicast don't implement thi hardware doesn't provide filtering based upon individual multicast addresses. Instead, they inspect using isUnwantedMulticastPacket: to weed out packets to unwanted multicast addresses. This meth enableMulticastMode, is invoked in the I/O thread every time a new multicast address is added to t maintains.

enableMulticastMode, isUnwantedMulticastPacket:, removeMulticastAddress:

(IONetwork *)attachToNetworkWithAddress:(enet_addr_t)address

Creates an IONetwork instance and attaches to the network subsystem by sending the IONetwork a initForNetworkDevice:... message. Before returning, this method logs a message including the ethe the IONetwork instance just created.

You invoke this method at the end of your implementation of initFromDeviceDescription:. You mu resetAndEnable:NO before invoking this method, as described under initFromDeviceDescription:.

(void)clearTimeout

If a transmission timeout is scheduled, unschedules the timeout. This method is normally invoked implementation of interruptOccurred.

(void)disableMulticastMode

Does nothing. Subclasses that support multicast mode and implement enableMulticastMode should method so that it disables the hardware's support for multicast mode. This method is invoked in th last multicast address has been removed from the list that IOEthernet maintains.

enableMulticastMode

(void)disablePromiscuousMode

Does nothing. Subclasses that support promiscuous mode must implement this method so that it di support for promiscuous mode. This method is invoked in the I/O thread by the networking subsys

enablePromiscuousMode

(BOOL)enableMulticastMode

Does nothing and returns YES. Subclasses that support promiscuous mode must implement this m
the hardware's support for promiscuous mode. This method is invoked in the I/O thread by the net

enablePromiscuousMode

free

Frees the IOEthernet instance and returns nil.

initFromDeviceDescription:(IODeviceDescription *)deviceDescription

Initializes a newly allocated IOEthernet instance. This includes invoking initFromDeviceDescripti
startIOThread setting the name, kind, and unit of this instance and invoking registerDevice.

Subclasses of IOEthernet should implement this method so that it invokes [super initFromDeviceD
performs any device-specific initialization. The subclass implementation should invoke resetAndE
finish by invoking attachToNetworkWithAddress:. An example of a subclass implementation of th
Italicized text delineated in angle brackets, that is << >>, is to be filled in with device-specific cod

(BOOL)isRunning

Returns YES if the hardware is currently capable of communication with other stations in the netw
NO.

setRunning:

(BOOL)isUnwantedMulticastPacket:(ether_header_t *)header

Determines whether the specified packet is to a multicast address that this device shouldn't listen t
packet should be dropped otherwise, returns NO.

(unsigned int)relativeTimeout

Returns the number of milliseconds until a transmission timeout will occur. If no transmission time
scheduled, this method returns zero.

(void)removeMulticastAddress:(enet_addr_t *)address

Does nothing. Subclasses that support multicast mode can implement this method so that it notifies
should stop listening for packets sent to address.

addMulticastAddress:, disableMulticastMode

(BOOL)resetAndEnable:(BOOL)enable

Does nothing and returns YES. Subclasses of IOEthernet must implement this method so that it res
hardware. Interrupts should be enabled if enable is YES otherwise, they should be left disabled. In
method should invoke setRunning: to record the basic state of the device.

This method should return YES if it encounters no errors (no matter what the value of enable is) if
should return NO. For example, the result from resetAndEnable:NO should be YES if the reset is s

The only time this method is invoked, with the exception of any invocations from your IOEthernet
implementation, is during initialization. Specifically, resetAndEnable:YES is invoked once in the I
attachToNetworkWithAddress: is invoked.

setRunning:

(void)setRelativeTimeout:(unsigned int)timeout

Schedules a timeout to occur in timeout milliseconds. This method is generally invoked by the IOE
method. When timeout milliseconds pass without the timeout being cleared (with clearTimeout), th
timeoutOccurred is invoked.

(void)setRunning:(BOOL)running

Sets whether the hardware is on line. The value of running should be YES to indicate that the hard
otherwise, it should be NO. This method is invoked only by methods in IOEthernet subclassesÐno
method implementations. You should invoke this method in your implementation of resetAndEnab

isRunning

This method can be invoked in many contexts, not just from the I/O thread (or from the I/O task). I
and interruptOccurred can run at the same time, so any common structures they both use must be p