free

Registering the class+ deviceStyle

Getting and setting the interrupt port

attachInterruptPort
interruptPort

Handling messages to the interrupt port

commandRequestOccurred
interruptOccurred
interruptOccurredAt:
receiveMsg
timeoutOccurred
waitForInterrupt:

Running an I/O thread startIOThread

startIOThreadWithPriority:
startIOThreadWithFixedPriority:

Getting and setting the IODeviceDescription

deviceDescription
setDeviceDescription:

initFromDeviceDescription:

Reserving I/O ports reservePortRange:

unmapMemoryRange:from:

Dealing with DMA channels enableChannel:
disableChannel:
reserveChannel:
releaseChannel:

Dealing with DMA buffers startDMAForBuffer:channel:
createDMABufferFor:length:read:needsLowMemory:
limitSize:
freeDMABuffer:
abortDMABuffer:

Setting the DMA mode setTransferMode:forChannel:
setAutoinitialize:forChannel:
setIncrementMode:forChannel:

Using the EISA extended mode register
setDMATransferWidth:forChannel:
setDMATiming:forChannel:
setEOPAsOutput:forChannel:
setStopRegisterMode:forChannel:

Getting a DMA channel's status currentAddressForChannel:
currentCountForChannel:
getDMATransferWidth:forChannel:
isDMADone:

Optional DMA locking reserveDMALock
releaseDMALock

Getting information about EISA slots
isEISAPresent
getEISAId:forSlot:

isPCIPresent

Reading and writing the entire configuration space
+ getPCIConfigSpace:withDeviceDescription:
+ setPCIConfigSpace:withDeviceDescription:
getPCIConfigSpace:withDeviceDescription:
setPCIConfigSpace:withDeviceDescription:

Reading and writing the configuration space
+ getPCIConfigData:atRegister:withDeviceDescription:
+ setPCIConfigData:atRegister:withDeviceDescription:
getPCIConfigData:atRegister:withDeviceDescription:
setPCIConfigData:atRegister:withDeviceDescription:

mapAttributeMemoryTo:findSpace:
unmapAttributeMemory:

(IOReturn)attachInterruptPort

Creates the interrupt port, if none exists already, and requests that the interrupt port receive all inte device's reserved interrupts. This method is invoked whenever an interrupt is enabled. Returns IO_ successful otherwise, returns IO_R_NOT_ATTACHED.

interruptPort, enableAllInterrupts (ªInstance Methods (ISA/EISA Architecture)º)

(void)commandRequestOccurred

Does nothing subclasses can implement this method if desired. This method is invoked by the defa (implemented by startIOThread...) whenever it receives a bodyless message with ID IO_COMMA a driver that handles user requests can use this message to notify the I/O thread that it should execu been placed in global data.

startIOThread

deviceDescription

Returns the IODeviceDescription associated with this instance.

setDeviceDescription:

free

Deallocates the IODirectDevice's memory and its interrupt port, if one exists. Returns nil.

(void)interruptOccurred

Invokes interruptOccurredAt: with an argument of zero. This method is invoked by the default I/O by startIOThread...) whenever it receives a bodyless Mach message with the ID IO_DEVICE_INT Subclasses that support only one interrupt should implement this method so that it processes the ha described in Chapter 1 and 2.

interruptOccurredAt:, startIOThread

(void)interruptOccurredAt:(int)localInterrupt

Does nothing subclasses that need to handle interrupts should implement this method so that it pro interrupt, as described in Chapter 1. This method is invoked by the default I/O thread (implemente whenever it receives a bodyless Mach message with an ID between IO_DEVICE_INTERRUPT_M IO_DEVICE_INTERRUPT_MSG_LAST (excluding IO_DEVICE_INTERRUPT_MSG).

interruptOccurred, startIOThread

(void)otherOccurred:(int)msgID

Does nothing subclasses can implement this method if desired. This method is invoked by the defa
(implemented by startIOThread...) whenever it receives a bodyless message with an unrecognized
msgID.

receiveMsg, startIOThread

(void)receiveMsg

Dequeues the next Mach message from the interrupt port and throws it away subclasses can implen
desired to handle custom messages. This method is invoked by the default I/O thread (implemente
whenever it tries to receive a message that has a body. To implement this message, you need to cal
interrupt port. In this sample implementation, fill in the italicized text between angle brackets, that
specific code:

otherOccurred:, startIOThread

(void)setDeviceDescription:deviceDescription

thread. This thread, which is appropriate for most drivers, sits in an endless loop that does the follo

·Waits for a Mach message on the interrupt port by invoking waitForInterrupt:

·If the message couldn't be dequeued because it was too large, invokes receiveMsg so that the sub
handle the message itself

·If the message is dequeued successfully, invokes one of five methods, depending on the message

startIOThreadWithFixedPriority:, startIOThreadWithPriority:

(IOReturn)startIOThreadWithFixedPriority:(int)priority

The same as startIOThreadWithPriority:, except that the I/O thread's priority never lessens due to a
you do performance tuning by disabling priority aging.

For more information about scheduling policies and priorities, see Chapter 1 of the NEXTSTEP O
Software manual.

startIOThread, startIOThreadWithPriority:

(IOReturn)startIOThreadWithPriority:(int)priority

The same as startIOThread, except that the I/O thread runs at the specified priority. This method le
tuning by raising or lowering the thread's scheduling priority. By default, kernel I/O threads start w
the maximum user priority (currently 18).

For more information about priorities, see Chapter 1 of the NEXTSTEP Operating System Softwa

startIOThread, startIOThreadWithFixedPriority:

(void)timeoutOccurred

Does nothing subclasses that support timeouts can implement this method. See the IOEthernet clas
implementing this method as part of timeout support. This method is invoked by the default I/O th
startIOThread...) whenever it receives a bodyless Mach message with an ID of IO_TIMEOUT_MS
IOSCSIController class for an example of sending Mach messages.

startIOThread

If a message is already on the queue when this method is invoked, this method dequeues the messa
give up the processor before returning. Without this precaution, a thread with many messages queu
kernel threads from being executed.

If this method successfully detects and dequeues a message, it sets msgId to the message's ID and
IO_R_SUCCESS.

startIOThread

(void)abortDMABuffer:(IOEISADMABuffer)buffer

Frees the memory allocated to buffer. If a read transfer is in progress, the data read is lost.

freeDMABuffer:

(IOEISADMABuffer)createDMABufferFor:(unsigned int *)physicalAddress length:(unsigned
read:(BOOL)isRead
needsLowMemory:(BOOL)lowerMem
limitSize:(BOOL)limitSize

Returns a DMA buffer for the contents of physical memory starting at physicalAddress and contin
bytes. You should specify YES for isRead if the data will be read from the device if the data will b
specify NO. lowerMem should be YES if the transfer must be from or to the first 16MB of physica
by some ISA devices) otherwise, it should be NO. To limit the size of the transfer to 64KB, specify
otherwise, limitSize should be NO.

This method changes the physical address if necessary to accommodate the ISA bus. When the phy
changed, the data is copied to the new physical address (if the transfer is a write), and the new phy
in physicalAddress.

Returns NULL if kernel memory for the buffer couldn't be allocated.

freeDMABuffer:

(unsigned int)currentAddressForChannel:(unsigned int)localChannel

Returns the physical address currently in the address register of the specified DMA channel. This r
at any timeÐeven when DMA is in progress. This method is often used along with autoinitialize m
help diagnose errors when a device or channel aborts a DMA transfer.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

currentCountForChannel:, setAutoinitialize:forChannel:

(unsigned int)currentCountForChannel:(unsigned int)localChannel

Returns the number of bytes remaining to be transferred on the specified channel. The maximum n

(void)disableAllInterrupts

Disables all interrupts associated with this IODirectDevice, so that no interrupts can be generated b
Returns IO_R_NO_INTERRUPT if no interrupt port is attached otherwise, returns IO_R_SUCCE

enableAllInterrupts, disableInterrupt:

(void)disableChannel:(unsigned int)localChannel

If the DMA channel corresponding to localChannel is reserved by this device, this method disables
typically disable the channel just before changing its setting. You need to invoke enableChannel: o
up so that transfers can occur.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

enableChannel:

(void)disableInterrupt:(unsigned int)localInterrupt

Disables the interrupt corresponding to localInterrupt.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

disableAllInterrupts, enableInterrupt:

(IOReturn)enableAllInterrupts

Creates and attaches an interrupt port, if one isn't already attached, and enables all interrupts assoc
IODirectDevice. Returns IO_R_NO_INTERRUPT if the interrupt port couldn't be attached otherw
IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

attachInterruptPort, disableAllInterrupts, enableInterrupt:

(IOReturn)enableChannel:(unsigned int)localChannel

Enables transfers on the DMA channel corresponding to localChannel. Returns IO_R_NOT_ATTA
doesn't correspond to a DMA channel or if the DMA channel isn't reserved by this device. Otherw
IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

disableChannel:, startDMAForBuffer:channel:

(void)freeDMABuffer:(IOEISADMABuffer)buffer

Completes the transfer associated with buffer and frees the buffer. buffer should be a value returne
createDMABufferFor:.... If createDMABufferFor:... changed the physical address and the transfer
moves the data from the new physical address to the old one. In other words, any data that's read a
passed to createDMABufferFor:... in the physicalAddress argument, not at the address returned in

abortDMABuffer:, createDMABufferFor:length:read:needsLowMemory:limitSize:

(IOReturn)getDMATransferWidth:(IOEISADMATransferWidth *)width forChannel:(unsigne

Returns in width the width currently used for DMA transfers on the specified channel. The width c
16-bit (IO_16BitByteCount), or 32-bit (IO_32Bit). On EISA systems, you can set the width using
forChannel:.

If localChannel doesn't correspond to a DMA channel, this method does nothing and returns IO_R
the DMA channel isn't reserved by this device, this method does nothing and returns IO_R_NOT_
Otherwise, this method returns IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

setDMATransferWidth:forChannel:

(BOOL)getEISAId:(unsigned int *)id forSlot:(int)slotNumber

Returns in id the EISA id for the specified slot. Returns YES if the slot is a valid EISA slot otherw
can use this method to loop through the computer's slots, testing each slot for whether it contains a
example, the following code is executed in the QVision display driver's initFromDeviceDescriptio
whether QVision hardware is present in the system.

isEISAPresent

If this method returns YES, interrupts from the device result directly in a call to handler, with the d
argument arg, at interrupt level ipl. Otherwise, interrupts result in a Mach message to the instance's

If you implement this method, you should use interrupt level 3 (IPLDEVICE, as defined in kernser
higher interrupt level is absolutely necessary. Using interrupt levels greater than 3 requires great ca
NeXT kernel internals.

initFromDeviceDescription:deviceDescription

Initializes and returns the IODirectDevice instance. Records deviceDescription as the IODeviceDe
to this IODirectDevice. Reserves all the interrupts, DMA channels, and I/O ports specified in devic
resources can't be reserved, releases all resources and returns nil.

This method must be invoked before any methods that require local equivalents of resources can b
mapMemoryRange:... requires that you specify the local equivalent of a memory range. However,
know what memory ranges they can use until initFromDeviceDescription: has been invoked. This
that subclass implementations of initFromDeviceDescription: must invoke the superclass's implem
initFromDeviceDescription: before they can map any memory ranges or do anything else that requ
resources.

(BOOL)isDMADone:(unsigned int)localChannel

Returns YES if DMA has completed on the specified channel otherwise, returns NO. If localChann
to a DMA channel, this method does nothing and returns IO_R_INVALID_ARG.

(IOReturn)mapMemoryRange:(unsigned int)localMemoryRange to:(vm_address_t *)destinatio
findSpace:(BOOL)findSpace
cache:(IOCache)caching

Maps the device memory corresponding to localMemoryRange into the calling task's address spac
is the local range number in the device description.

If findSpace is TRUE, this method ignores the destinationAddress and determines where the mapp
returning the value in destinationAddress. If findSpace is FALSE, this method truncates destinatio
page boundary, maps the memory to the truncated address, and returns the truncated address.

The caching argument determines how the memory is cached. Usually, it should be IO_WriteThro
caching seems to be causing problems, try using IO_CacheOff instead.

If localMemoryRange doesn't correspond to one of this device's memory ranges, IO_R_INVALID
There must also be more than one I/O port range associated with the device (i.e. [deviceDescriptio
otherwise IO_R_INVALID_ARG is returned. If the mapping couldn't be performed for another re
IO_R_NO_SPACE is returned. If the mapping was successful, returns IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

unmapMemoryRange:from:

(void)releaseChannel:(unsigned int)localChannel

Releases the DMA channel corresponding to localChannel so that another device can use the chan

reserveChannel:

(void)releaseDMALock

Releases the lock associated with DMA. This method panics if this IODirectDevice doesn't hold th

Most drivers don't need to use DMA locking. However, the floppy drive (and possibly other devic
underruns when the bus is saturated. As a result, the floppy driver and drivers for devices that tend
DMA locking to avoid performing I/O at the same time. DMA locking is ignored by all other devi

You don't have to use DMA locking unless your device is having DMA underruns or is causing an
underruns. Sometimes these underruns occur on ISA computers, but not EISA ones. If your device
drive to have underruns, you'll see the following error on the console while your device is perform

reserveDMALock

(void)releaseInterrupt:(unsigned int)localInterrupt

Releases the interrupt corresponding to localInterrupt so that another device can use the interrupt.

reserveInterrupt:

(IOReturn)reserveChannel:(unsigned int)localChannel

Reserves the DMA channel corresponding to localChannel so that no other device can use the chan IO_R_NOT_ATTACHED if localChannel doesn't correspond to a DMA channel or if the DMA ch another device. Otherwise, returns IO_R_SUCCESS.

You don't normally have to invoke this method, since initFromDeviceDescription: reserves all the channels.

releaseChannel:

(void)reserveDMALock

Reserves the lock associated with DMA. See releaseDMALock for information on DMA locking.

(IOReturn)reserveInterrupt:(unsigned int)localInterrupt

Reserves the interrupt corresponding to localInterrupt so that no other device can use it. Returns IO_R_NOT_ATTACHED if localInterrupt doesn't correspond to an interrupt or if another device interrupt. Otherwise, returns IO_R_SUCCESS.

You don't normally have to invoke this method, since initFromDeviceDescription: reserves all the

releaseInterrupt:

(IOReturn)reservePortRange:(unsigned int)localPortRange

Releases the range of I/O ports corresponding to localPortRange and returns IO_R_SUCCESS.

You don't normally have to invoke this method, since initFromDeviceDescription: reserves all the

releasePortRange:

(IOReturn)setAutoinitialize:(BOOL)flag forChannel:(unsigned int)localChannel

Sets the specified channel's autoinitialize DMA mode to on if flag is YES otherwise, sets it off. Th mode stays in effect until this method is invoked again or the computer is rebooted. By default, aut disabled.

If localChannel doesn't correspond to a DMA channel, this method does nothing and returns IO_R the DMA channel isn't reserved by this device, this method does nothing and returns IO_R_NOT_ Otherwise, this method returns IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego initFromDeviceDescription: is invoked.

setIncrementMode:forChannel:, setTransferMode:forChannel:

Because this method uses a local equivalent of a resource, it can't be invoked until after this categ
initFromDeviceDescription: is invoked.

(IOReturn)setDMATransferWidth:(IOEISADMATransferWidth)width forChannel:(unsigned

Makes the specified channel use the specified width for DMA transfers. The width can be 8-bit (IO
(IO_16BitByteCount), or 32-bit (IO_32Bit). The 16-bit mode requires byte counting, not word cou
unsupported). This method is valid only on EISA systems.

If the system is ISA-based, this method does nothing and returns IO_R_UNSUPPORTED. If local
correspond to a DMA channel, this method does nothing and returns IO_R_INVALID_ARG. If the
reserved by this device, this method does nothing and returns IO_R_NOT_ATTACHED. Otherwis
IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this categ
initFromDeviceDescription: is invoked.

(IOReturn)setEOPAsOutput:(BOOL)flag forChannel:(unsigned int)localChannel

Selects whether the specified channel's EOP pin is an output signal (the default) or an input signal
only on EISA systems.

If the system is ISA-based, this method does nothing and returns IO_R_UNSUPPORTED. If local
correspond to a DMA channel, this method does nothing and returns IO_R_INVALID_ARG. If the
reserved by this device, this method does nothing and returns IO_R_NOT_ATTACHED. Otherwis
IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this categ
initFromDeviceDescription: is invoked.

(IOReturn)setIncrementMode:(IOIncrementMode)mode forChannel:(unsigned int)localChann

This method lets the driver specify how the start address and length of its DMA buffers should be
the increment mode is IO_Increment, so each DMA buffer is interpreted so that if the start address
m, the data in addresses n through n + m 1 are transferred. By setting the increment mode to IO_D
driver specifies that the affected addresses should be n through n m + 1. The new increment mode
method is invoked again or until the computer is rebooted.

If localChannel doesn't correspond to a DMA channel, this method does nothing and returns IO_R
the DMA channel isn't reserved by this device, this method does nothing and returns IO_R_NOT_
Otherwise, this method returns IO_R_SUCCESS.

setAutoinitialize:forChannel:, setTransferMode:forChannel:

(IOReturn)setStopRegisterMode:(IOEISAStopRegisterMode)mode forChannel:(unsigned int)l

(IOReturn)setTransferMode:(IODMATransferMode)mode forChannel:(unsigned int)localCha

Sets the specified channel's transfer mode to mode. The new transfer mode stays in effect until thi
again or the computer is rebooted.

If localChannel doesn't correspond to a DMA channel, this method does nothing and returns IO_R
the DMA channel isn't reserved by this device, this method does nothing and returns IO_R_NOT_
Otherwise, this method returns IO_R_SUCCESS.

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

setAutoinitialize:forChannel:, setIncrementMode:forChannel:

(IOReturn)startDMAForBuffer:(IOEISADMABuffer)buffer channel:(unsigned int)localChann

Begins DMA with buffer on the DMA channel specified by localChannel, and returns IO_R_SUC
started if localChannel doesn't correspond to a DMA channel (in which case IO_R_INVALID_AR
DMA channel isn't assigned, or if no DMA frames could be allocated (IO_R_NO_FRAMES is ret

Because this method uses a local equivalent of a resource, it can't be invoked until after this catego
initFromDeviceDescription: is invoked.

(void)unmapMemoryRange:(unsigned int)localMemoryRange
from:(vm_address_t)address

Unmaps the device memory corresponding to localMemoryRange from the calling task's address s
address must be the same as the value returned by the destinationAddress argument of mapMemor
cache: for the same localMemoryRange.

mapMemoryRange:to:findSpace:cache:

(IOReturn)getPCIConfigData:(unsigned long *)data
atRegister:(unsigned char)address

Reads from the device's configuration space at the byte address address. All accesses are 32 bits w
be aligned as such.

(IOReturn)getPCIConfigSpace:(IOPCIConfigSpace *)configurationSpace

Reads the device's entire configuration space. Returns IO_R_SUCCESS if successful. If this meth
should make no assumptions about the state of the data returned in the IOPCIConfigSpace struct.

(BOOL)isPCIPresent

Returns YES if PCI Bus support is enabled. Returns NO otherwise.

(IOReturn)setPCIConfigData:(unsigned long)data
atRegister:(unsigned char)address

Writes to the device's configuration space at the byte address address. All accesses are 32 bits wid
aligned as such.

(IOReturn)setPCIConfigSpace:(IOPCIConfigSpace *)configurationSpace

Writes the device's entire configuration space. Returns IO_R_SUCCESS if successful. If this meth
should make no assumptions about the state of the device's configuration space.

(IOReturn)mapAttributeMemoryTo:(vm_address_t *)destinationAddressfindSpace:(BOOL)fi

Maps attribute memory to destinationAddress in findSpace.

unmapAttributeMemory: