initFromDeviceDescription:	
	free
	reset
Starting and stopping DMA	startDMAForChannel:read:buffer:
	bufferSizeForInterrupts:
	stopDMAForChannel:read:
Getting DMA buffer address and size	

	isOutputActive
Getting supported sampling rates	acceptsContinuousSamplingRates
	getSamplingRates:count:
	getSamplingRatesLow:high:
Getting supported data encodings	getDataEncodings:count:
Getting device settings	channelCount
	dataEncoding
	sampleRate
Determining what hardware settings are or should be	inputGainLeft
	inputGainRight
	isOutputMuted
	isLoudnessEnhanced
	outputAttenuationLeft
	outputAttenuationRight
Setting hardware state	updateInputGainLeft
	updateInputGainRight
	updateOutputMute
	updateLoudnessEnhanced
	updateOutputAttenuationLeft
	updateOutputAttenuationRight

(BOOL)acceptsContinuousSamplingRates

Returns NO. Drivers that accept continuous sampling rates, as opposed to accepting a few, discret
implement this method so that it returns YES. For example, if a device has a low rate of 2000 Hz a
Hz and supports every sampling rate in between, its implementation of this method should return Y

getSamplingRates:, getSamplingRatesLow:High:

(unsigned int)channelCount

Returns the number of sound channels to be used for the audio data that's about to be played or rec
which can be either 1 (for mono) or 2 (for stereo), is determined during mixing and is set before sta
is invoked.

dataEncoding, sampleRate

(unsigned int)channelCountLimit

Returns zero. Drivers must implement this method so that it returns either 1 (if only mono is suppo
mono and stereo are supported).

NX_SoundStreamDataEncoding_Linear8, NX_SoundStreamDataEncoding_Mulaw8, and
NX_SoundStreamDataEncoding_Alaw8.

channelCount, sampleRate

free

Frees the instance and returns nil.

(void)getDataEncodings:(NXSoundParameterTag *)encodings count:(unsigned int *)numEnco

Returns zero in numEncodings. Subclasses must override this method to supply an array of suppor
Possible values (defined in the header file soundkit/NXSoundParameterTags.h) are currently
NX_SoundStreamDataEncoding_Linear16, NX_SoundStreamDataEncoding_Linear8,
NX_SoundStreamDataEncoding_Mulaw8, and NX_SoundStreamDataEncoding_Alaw8. Below is
implementing this method. Note that you don't have to allocate memory for encodings it already h
all possible encodings.

(void)getInputChannelBuffer:(void *)address size:(unsigned int *)byteCount

Gets the starting address and size of the (already allocated) DMA buffer for the input channel. This
driver to access data in the audio buffer directly.

getOutputChannelBuffer:size:

(void)getOutputChannelBuffer:(void *)address size:(unsigned int *)byteCount

Gets the starting address and size of the (already allocated) DMA buffer for the output channel. Th
driver to access data in the audio buffer directly.

getInputChannelBuffer:size:

(void)getSamplingRates:(int *)rates count:(unsigned int *)numRates

Returns zero in numRates. Subclasses must override this method to supply the supported sampling
which has room for up to 256 entries. If the driver supports continuous sampling rates, this method
common sampling rates, as shown below.

(void)getSamplingRatesLow:(int *)lowRate high:(int *)highRate

Returns zero in lowRate and highRate. Subclasses must override this method to supply their highes
sampling rates. Here's an example of implementing this method.

acceptsContinuousSamplingRates, getSamplingRates:

initFromDeviceDescription:description

Initializes a newly allocated IOAudio instance. Subclasses don't generally override this method the
their probe: method. Subclasses perform device-specific initialization in their implementation of th

IOAudio's implementation of initFromDeviceDescription: invokes super's version of initFromDev
invokes attachInterruptPort, sets the interrupt port to have a maximum backlog, and then performs
it creates and initializes the private objects that perform much of the driver's work, creates private
to listen to requests on the ports. Finally, it invokes registerDevice. Returns nil if initialization was
otherwise, returns the IOAudio instance.

(unsigned int)inputGainLeft

Returns the general scaling factor that's applied to the left channel of the incoming sound. This val
from 0 to 32768, where 0 is no gain and 32768 is maximum gain. User-level programs specify the
Kit. To support input gain, you must implement updateInputGainLeft and updateInputGainRight.

inputGainRight

(unsigned int)inputGainRight

Returns the general scaling factor that's applied to the right channel of the incoming sound. This v
from 0 to 32768, where 0 is no gain and 32768 is maximum gain. User-level programs specify the
Kit. To support input gain, you must implement updateInputGainLeft and updateInputGainRight.

inputGainLeft

(IOAudioInterruptClearFunc)interruptClearFunc

Does nothing and returns zero. Subclasses must implement this method so that it returns the addres
clears interrupts on the card. The function is called only when the audio system needs to guarantee
pending interrupts. If you don't implement this method and function, your card is likely to suffer fr
with some applications. The function runs at interrupt level, so it must not block.

(void)interruptOccurredForInput:(BOOL *)serviceInput
    forOutput:(BOOL *)serviceOutput

Notifies the instance that an interrupt occurred for its hardware. The IOAudio version of this meth
message each subclass must implement this method.

The subclass implementation of this method should try to determine whether the hardware really h
this method should clear the card's interrupt state, set serviceInput to YES if the interrupt was for i
serviceOutput to YES if the interrupt was for output. (The values of serviceInput and serviceOutpu
)

After invoking this method, IOAudio checks whether any more data is available for DMA on the c
service. If none is available, stopDMAForChannel:read: is invoked. IOAudio always invokes this
thread.

(BOOL)isInputActive

Returns YES if data is being read from the hardware using DMA otherwise, returns NO.

isOutputActive

(BOOL)isLoudnessEnhanced

Returns YES if loudness is enhanced otherwise, returns NO. Loudness enhancement refers to the a
hardware to help compensate for the decreased sensitivity of the human ear by boosting the gain at
frequencies as the volume is decreased. User-level programs specify whether to use loudness enha
NX_SoundDeviceOutputLoudness parameter. To support loudness enhancement, you must impleme
updateLoudnessEnhanced.

(BOOL)isOutputActive

Returns YES if data is being sent to the hardware using DMA otherwise, returns NO.

isInputActive

(BOOL)isOutputMuted

Returns YES if output is muted otherwise, returns NO. The user can mute audio output by holding
key and pressing the Delete key. User-level programs can mute output using the Sound Kit.

updateOutputMute

(int)outputAttenuationLeft

(int)outputAttenuationRight

Returns the attenuation setting of the right channel of the device. The user modifies the left and rig
simultaneously using the Volume slider in the Preferences application or with the Insert and Delete
User-level programs can specify the attenuation using the Sound Kit. The range is -84 decibels (in
(no attenuation).

updateOutputAttenuationRight, outputAttenuationLeft

(BOOL)reset

Generates an error message and returns NO. Subclasses must implement this method so that it rese
hardware. This method is invoked from initFromDeviceDescription:, as described above.

This method should initialize basic information by invoking setName: and setDeviceKind:. It shou
its interrupt (IRQ) and DMA channels (all obtained from its IODeviceDescription) have valid valu
hardware, this method should disable its DMA channels and then set any DMA parameters necess
width.

This method should return YES on success otherwise, it should return NO, which will cause initFr
to return nil.

initFromDeviceDescription:, setName: (IODevice), setDeviceKind: (IODevice)

(unsigned int)sampleRate

Returns the sample rate to be used for the audio data that's about to be played or recorded. This va
during mixing and is set before startDMAForChannel:... is invoked.

channelCount, dataEncoding

(BOOL)startDMAForChannel:(unsigned int)localChannel
read:(BOOL)isRead
buffer:(IODMABuffer)buffer
bufferSizeForInterrupts:(unsigned int)bufferSize

Generates an error message and returns NO. Subclasses must override this method.

This method should perform DMA after configuring the hardware to reflect the values returned by
dataEncoding, and channelCount. The DMA should be set up so that it generates an interrupt after
interval. If isRead is YES, then the DMA is from the card to memory otherwise, DMA is from mer
the example IOAudio driver for an implementation of this method.

IOAudio invokes this method from the I/O thread. You should never invoke this method in an IOA
implementation.

This method should return YES if it started DMA successfully otherwise, it should return NO.

startDMAForBuffer:channel (IODirectDevice architecture-specific category), enableChannel (IO
architecture-specific category), enableAllInterrupts (IODirectDevice architecture-specific categor

(void)stopDMAForChannel:(unsigned int)localChannel read:(BOOL)isRead

startDMAForChannel:read:buffer:bufferSizeForInterrupts:, disableChannel (IODirectDevice arch
category), disableAllInterrupts (IODirectDevice architecture-specific category)

(void)timeoutOccurred

Notifies the instance that although a DMA transaction is in progress, no interrupts have been detec
(currently one second). The IOAudio version of this method does nothing each subclass can implem

The subclass implementation of this method might reset the hardware. IOAudio invokes this metho

(void)updateInputGainLeft

Does nothing. Subclasses should implement this method so that it updates the hardware to the valu
inputGainLeft. You generally have to convert the device-independent value returned by inputGainL
value for your device.

updateInputGainRight

(void)updateInputGainRight

Does nothing. Subclasses should implement this method so that it updates the hardware to match th
inputGainRight. You generally have to convert the device-independent value returned by inputGai
appropriate value for your device. IOAudio invokes this method from the I/O thread.

updateInputGainLeft

(void)updateLoudnessEnhanced

Does nothing. Subclasses that support loudness enhancement should implement this method so tha
hardware to match the value returned by isLoudnessEnhanced. IOAudio invokes this method from

(void)updateOutputAttenuationLeft

Does nothing. Subclasses should implement this method so that it updates the hardware to match th
outputAttenuationLeft. You generally have to convert the device-independent value returned by ou
the appropriate value for your device. Here's an example of implementing this method.

updateOutputAttenuationRight

(void)updateOutputAttenuationRight

Does nothing. Subclasses should implement this method so that it updates the hardware to match th
outputAttenuationRight. You generally have to convert the device-independent value returned by o
to the appropriate value for your device. IOAudio invokes this method from the I/O thread.

updateOutputAttenuationLeft

(void)updateOutputMute

Does nothing. Subclasses should implement this method so that it mutes the output if isOutputMut
unmutes the output if isOutputMuted returns NO. IOAudio invokes this method from the I/O threa