init

initFrame:
initFrame:icon:tag:target:action:key:enabled:
initFrame:title:tag:target:action:key:enabled:

Setting the Button type setType:

Setting the state setState:

state

Setting the repeat interval setPeriodicDelay:andInterval:

getPeriodicDelay:andInterval:

Setting the titles setTitle:

setTitleNoCopy:
title
setAltTitle:
altTitle

Setting the icons setIcon:

setIcon:position:
icon
setAltIcon:
altIcon
setImage:
image
setAltImage:
altImage
setIconPosition:
iconPosition

Modifying graphic attributes setTransparent:

isTransparent
setBordered:
isBordered

Displaying the Button display

highlight:

Setting the key equivalent setKeyEquivalent:

(BOOL)acceptsFirstMouse

Returns YES. Buttons always accept the mouse-down event that activates a Window, regardless o
enabled.

(const char *)altIcon

Returns the name of the NXImage that appears on the Button when it's in its alternate state, or NU
alternate icon or the NXImage has no name. This NXImage is displayed only for Buttons that high
alternate state by displaying their alternate contents (as opposed to simply lighting or pushing in).

setAltIcon:, setIconPosition:, altImage, icon, image, setType:

altImage

Returns the NXImage that appears on the Button when it's in its alternate state, or nil if there is no
This Button only displays its alternate NXImage if it highlights or shows its alternate state by displ
contents.

setAltImage:, setIconPosition:, altIcon, image, icon, setType:

(const char *)altTitle

Returns the string that appears on the Button when it's in its alternate state, or NULL if there isn't
is only displayed if the Button highlights or shows its alternate state by displaying its alternate con

setAltTitle:, title, setType:

display

Displays the Button. This method is overridden from View so that displayFromOpaqueAncestor::
Button is not opaque. Returns self.

isOpaque (Cell), isTransparent, setTransparent:

highlight:(BOOL)flag

If the highlight state of the cell is not equal to flag, the Button is highlighted and the highlight state flag. Highlighting may involve the Button appearing ªpushed inº to the screen, displaying its alter lighting. This method issues a flushWindow message after highlighting the Button. Returns self.

setType:

(const char *)icon

Returns the name of the NXImage that appears on the Button when it's in its normal state, or NUL NXImage or the NXImage doesn't have a name. A Button that doesn't display its alternate conten its alternate state will always display its normal icon.

setIcon:, setIcon:position:, setIconPosition:, image, altIcon, altImage, setType:

(int)iconPosition

Returns the position of the icon (if any) on the Button. See setIconPosition: for the list of positions

setIconPosition:, setIcon:position:

image

Returns the NXImage that appears on the Button when it's in its normal state, or nil if there is no s NXImage is always displayed on a Button that doesn't change its contents when highlighting or sh state.

setImage:, setIconPosition:, icon, altImage, altIcon, setType:

init

Initializes and returns the receiver, a new Button instance, with a frame origin of (0, 0) and width a each. The new instance is enabled and displays the default title ªButtonº centered in its frame, but target, action, or key equivalent associated with it. The new Button is bordered, and is of type NX_MOMENTARYPUSH. One of the more specific initializers is usually used to initialize a But

initFrame:title:tag:target:action:key:enabled:, initFrame:icon:tag:target:action:key:enabled:, initF

initFrame:(const NXRect *)frameRect

Initializes and returns the receiver, a new Button instance, with default parameters in the given fra is enabled and displays the default title ªButtonº centered in its frame, but has no icon, tag, target, equivalent. The new Button is bordered, and is of type NX_MOMENTARYPUSH. One of the m is usually used to initialize a Button.

initFrame:title:tag:target:action:key:enabled:, initFrame:icon:tag:target:action:key:enabled:, initF

enabled:(BOOL)flag

Initializes and returns the receiver, a new Button instance that displays an icon. frameRect is the r
Button's position and size in its superview. iconName is the name of an NXImage that will be use
anInt is set as the Button's tag. anObject is set as the target, which will be sent aSelector when the
pressed. charCode is the new Button's key equivalent. flag determines whether the Button is enab
Button is bordered, and is of type NX_MOMENTARYPUSH.

This method is the designated initializer for Buttons that display icons. A Button that displays an i
to also display a title with the setTitle: and setIconPosition: methods.

setTitle:, setIconPosition:, setType:

initFrame:(const NXRect *)frameRect
title:(const char *)aString
tag:(int)anInt
target:anObject
action:(SEL)aSelector
key:(unsigned short)charCode
enabled:(BOOL)flag

Initializes and returns the receiver, a new Button instance that displays a text string. The argument
method are identical to those of initFrame:icon:tag:target:action:key:enabled:, except that aString i
Button will display instead of the name of an icon. The new Button is bordered, and is of type
NX_MOMENTARYPUSH.

This method is the designated initializer for Buttons that display text. A Button that displays an ic
also display an icon with the setIcon:position: method, or a combination of setIcon: or setImage: a

setIcon:, setImage:, setIconPosition:, setType:

(BOOL)isBordered

Returns YES if the Button has a border, NO otherwise. A Button's border isn't the single line of n
borders instead, it's a raised bezel ("bezel" usually refers to a depressed bezel, as seen on FormCel
shouldn't use the setBezeled: method with a Button.

setBordered:

(BOOL)isTransparent

Returns YES if the Button is transparent, NO otherwise. A transparent Button never draws itself, l
down events and tracks the mouse properly.

setTransparent:

(unsigned short)keyEquivalent

Returns the key equivalent character of the Button, or 0 if one hasn't been defined.

setKeyEquivalent:, performKeyEquivalent:

(BOOL)performKeyEquivalent:(NXEvent *)theEvent

If the character in theEvent matches the Button's key equivalent, simulates the user clicking the Bu
performClick: to self, and returns YES. Otherwise, does nothing and returns NO.

The Button won't perform the key equivalent if there's a modal panel present that the Button isn't

keyEquivalent, performClick:

setAltIcon:(const char *)iconName

Sets the Button's alternate icon by name iconName is the name of the NXImage to be displayed. I
Button even if autodisplay is on. Returns self.

A Button's alternate icon is only displayed if the Button highlights or shows its alternate state by c

altIcon, setIconPosition:, setAltImage:, setIcon:, setImage:, + findImageNamed: (NXImage), s
setAutodisplay: (View)

setAltImage:altImage

Sets the Button's alternate icon by id altImage is the NXImage to be displayed. Returns self.

A Button displays its alternate NXImage only if it highlights or displays its alternate state by using

altImage, setIconPosition:, setAltIcon:, setImage:, setIcon:, setType:

setAltTitle:(const char *)aString

Sets the title that the Button displays in its alternate state to aString. Returns self.

The alternate title is shown only if the Button changes its contents when highlighting or displaying

altTitle:, setTitle:, setType:

setBordered:(BOOL)flag

If flag is YES, the Button displays a border if NO, the Button doesn't display a border. A Button's
single line or most other Controls' borders instead, it's a raised bezel (ªbezelº usually refers to a d
on FormCells, for example). This method redraws the Button if the bordered state changes. Retur

isBordered

setIcon:(const char *)iconName

Sets the Button's icon by name iconName is the name of the NXImage to be displayed. Redraws t
returns self.

Combines setIcon: and setIconPosition: into one message. Returns self.

setIcon:, setIconPosition:

setIconPosition:(int)aPosition

Sets the position of the icon when a Button simultaneously displays both text and an icon. aPositio
following constants:

NX_TITLEONLYtitle only (no icon on the Button)
NX_ICONONLYicon only (no text on the Button)
NX_ICONLEFTicon is to the left of the text
NX_ICONRIGHTicon is to the right of the text
NX_ICONBELOWicon is below the text
NX_ICONABOVEicon is above the text
NX_ICONOVERLAPSicon and text overlap (text drawn over icon)

If the position is top or bottom, the alignment of the text will be changed to NX_CENTERED. Th
overridden with a subsequent setAlignment: method. Redraws the Button's inside and returns self

iconPosition, setIcon:position:, setAlignment: (Control)

setImage:image

Sets the Button's icon by id image is the NXImage to be displayed. Redraws the Button's inside a

A Button's icon is displayed when the Button is in its normal state, or all the time for a Button that
contents when highlighting or displaying its alternate state.

image, setIconPosition:, setIcon:, setAltImage:, setAltIcon:, setType:

setKeyEquivalent:(unsigned short)charCode

Sets the key equivalent character of the Button, and redraws the Button's inside if there is no icon
the Button. The key equivalent isn't displayed if the icon position is set to NX_TITLEONLY, NX
NX_ICONOVERLAPS that is, the Button must display both its title and its ªiconº (the key equiva
they must not overlap. Returns self.

To display a key equivalent on a Button, set the image and alternate image to nil, then set the key e
the icon position.

keyEquivalent, setIconPosition:, performKeyEquivalent:, setImage:, setAltImage:

setPeriodicDelay:(float)delay andInterval:(float)interval

Sets the message delay and interval for the Button. These two values are used if the Button is conf
setContinuous: message) to continuously send the action message to the target object while trackin
the amount of time (in seconds) that a continuous Button will pause before starting to periodically
to the target object. interval is the amount of time (also in seconds) between those messages. Retu

The maximum value allowed for both the delay and the interval is 60.0 seconds.

setState:(int)anInt

Sets the Button's state to anInt and redraws the Button. 0 is the normal or ªoffº state, and any non
alternate or ªonº state. Returns self.

state

setTitle:(const char *)aString

Sets the title displayed by the Button when in its normal state to aString. This title is always show
use their alternate contents when highighting or displaying their alternate state. Redraws the Butto
self.

setTitleNoCopy:, title, setAltTitle:, setType:

setTitleNoCopy:(const char *)aString

Similar to setTitle: but doesn't make a copy of aString. Returns self.

setTitle:

setTransparent:(BOOL)flag

Sets whether the Button is transparent, and redraws the Button if flag is NO. Returns self.

A transparent Button tracks the mouse and sends its action, but doesn't draw. A transparent Butto
sensitizing an area on the screen so that an action gets sent to a target when the area receives a mou

isTransparent

setType:(int)aType

Sets the way the Button highlights while pressed, and how it shows its state. Redraws the Button a
types available are for the most common Button types, which are also accessible in Interface Build
different behavior with ButtonCell's setHighlightsBy: and setShowsStateBy: methods. aType can
constants:

NX_MOMENTARYPUSH (the default): While the Button is held down it's shown as lit, and
screen if the Button is bordered. This type of Button is best for simply triggering actions, as it
always displays its normal icon or title. This option is called ªMomentary Pushº in Interface B
Inspector.

NX_MOMENTARYCHANGE: While the Button is pressed, the alternate icon or alternate titl
type always displays its normal title or icon (that is, it doesn't display its state). The miniaturiz
title bar is a good example of this type of Button. This option is called ªMomentary Changeº i
Button Inspector.

NX_SWITCH: A variant of NX_TOGGLE that has no border, and that has a default icon calle
alternate icon called ªswitchHº (these are identical to the ªNXswitchº and ªNXswitchHº syste
of Button is available as a separate palette item in Interface Builder.

NX_RADIOBUTTON: Like NX_SWITCH, but the default icon is ªradioº and the alternate ic
(identical to the ªNXradioº and ªNXradioHº system bitmaps). This type of Button is available
item in Interface Builder.

There is no constant for Interface Builder's ªMomentary Lightº type you can set this programmati

setType: (ButtonCell), setHighlightsBy: (ButtonCell), setShowsStateBy: (ButtonCell)

sound

Returns the Sound played when the Button is pressed, and whenever the cursor re-enters the Butto

setSound:

(int)state

Returns the Button's state, either 0 for normal or ªoff,º or 1 for alternate or ªon.º

setState:

(const char *)title

Returns the title displayed on the Button when it's in its normal state, or always if the Button doesn
contents for highlighting or displaying the alternate state. Returns NULL if there is no title.

setTitle:, altTitle, setType: