init
initTextCell:
initIconCell:
copyFromZone:
free

Determining component sizes calcCellSize:inRect:
getDrawRect:
getTitleRect:
getIconRect:

Setting the titles setTitle:
setTitleNoCopy:
title
setAltTitle:
altTitle
setFont:

Setting the icons setIcon:
icon
setAltIcon:
altIcon
setImage:
image
setAltImage:
altImage
setIconPosition:
iconPosition

Setting the Sound setSound:
sound

Setting the state setDoubleValue:

getPeriodicDelay:andInterval:

Tracking the mouse trackMouse:inRect:ofView:

Setting the key equivalent setKeyEquivalent:
setKeyEquivalentFont:
setKeyEquivalentFont:size:
keyEquivalent

Setting parameters setParameter:to:
getParameter:

Modifying graphic attributes setBordered:
isBordered
setTransparent:
isTransparent
isOpaque

Modifying display behavior setType:
setHighlightsBy:
highlightsBy
setShowsStateBy:
showsStateBy

Simulating a click performClick:

Displaying the ButtonCell drawInside:inView:
drawSelf:inView:
highlight:inView:lit:

Archiving read:
write:

(const char *)altIcon

Returns the name of the NXImage that appears on the ButtonCell when it's in its alternate state, or alternate icon or the NXImage has no name. This NXImage is displayed only for ButtonCells that 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 ButtonCell when it's in its alternate state, or nil if there i NXImage. This ButtonCell only displays its alternate NXImage if it highlights or shows its alterna its alternate contents.

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

(const char *)altTitle

Returns the string that appears on the ButtonCell when it's in its alternate state, or NULL if there i title is only displayed if the ButtonCell highlights or shows its alternate state by displaying its alter

getDrawRect:, getIconRect:, getTitleRect:

copyFromZone:(NXZone *)zone

Allocates, initializes, and returns a copy of the receiving ButtonCell. The copy is allocated from z
same data as the receiver.

(double)doubleValue

Returns 0.0 if the ButtonCell is in its normal state, 1.0 if the it's in its alternate state.

setDoubleValue:, floatValue, intValue, stringValue

drawInside:(const NXRect *)aRect inView:controlView

Draws the inside of the ButtonCell (the title, icon, and their background, but not the border) in aRe
aRect should be the same rectangle passed to drawSelf:inView:. The PostScript focus must be loc
when this message is sent. This method is invoked by drawSelf:inView: and by the Control classe
method. It's provided so that when a ButtonCell's state is set (with setIntValue:, for example), a m
ButtonCell's visual appearance can occur. Returns self.

If you subclass ButtonCell and override drawSelf:inView:, you must also override this method. H
override only this method and not drawSelf:inView: if your subclass doesn't draw outside the area
in.

drawInside:inView: (Cell), drawSelf:inView:, lockFocus (View)

drawSelf:(const NXRect *)cellFrame inView:controlView

Displays the ButtonCell in cellFrame within controlView. The PostScript focus must be locked or
message is sent. Draws the border of the ButtonCell if necessary, then invokes drawInside:inView

drawInside:inView:, lockFocus (View)

(float)floatValue

Returns 0.0 if the ButtonCell is in its normal state, 1.0 if it's in its alternate state.

setFloatValue:, doubleValue, intValue, stringValue

free

Frees the memory used by the ButtonCell and returns nil.

getIconRect:(NXRect *)theRect

Returns self and, by reference in theRect, the bounds of the area into which the icon of the ButtonC
will be the larger of the bounds for the normal and the alternate icons. If the ButtonCell has no ico
completely zeroed. You must pass the bounds of the ButtonCell in theRect (the same bounds calcu
inRect: and passed to drawSelf:inView:). This method assumes that the ButtonCell is being drawn
Returns self.

getTitleRect:, getDrawRect:, calcCellSize:inRect:

(int)getParameter:(int)aParameter

Returns the value of one of the frequently accessed flags for a ButtonCell. See setParameter:to: fo
parameters and corresponding methods. Since the parameters are also accessible through normal c
shouldn't need to use this method often.

setParameter:to:

getPeriodicDelay:(float *)delay andInterval:(float *)interval

Returns self, and by reference the delay and interval periods for a continuous ButtonCell. delay is
seconds) that a continuous ButtonCell will pause before starting to periodically send action messag
interval is the amount of time (also in seconds) between those messages.

setContinuous: (Cell), setPeriodicDelay:andInterval:

getTitleRect:(NXRect *)theRect

Returns self and, by reference in theRect a copy of the bounds of the area into which the ButtonCe
This will be the larger of the bounds for the normal and the alternate titles. If the ButtonCell has n
will be completely zeroed. You must pass the bounds of the ButtonCell in theRect (the same boun
calcCellSize:inRect: and passed to drawSelf:inView:). This method assumes that the ButtonCell is
flipped View.

getIconRect:, getDrawRect:, calcCellSize:inRect:

highlight:(const NXRect *)cellFrame
inView:controlView
lit:(BOOL)flag

Displays the ButtonCell in cellFrame if its highlight state is not equal to flag. The PostScript focus
controlView when this method is invoked. If flag is YES, the ButtonCell is displayed as highlight
depends on how the ButtonCell has been configured see the description of setHighlightsBy: for the
highlighting. This method does nothing if the ButtonCell is disabled or transparent. Returns self.

lockFocus (View)

(const char *)icon

Returns the name of the NXImage that appears on the ButtonCell when it's in its normal state, or N such NXImage or the NXImage doesn't have a name. A ButtonCell that doesn't display its alterna or show its alternate state will always display its normal icon.

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

(int)iconPosition

Returns the position of the ButtonCell's icon (if any). See setIconPosition: for a list of the valid po

setIconPosition:

image

Returns the NXImage that appears on the ButtonCell when it's in its normal state, or nil if there is NXImage is always displayed on a ButtonCell that doesn't change its contents when highlighting c state.

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

init

Initializes and returns the receiver, a new text ButtonCell, with the title ªButtonº aligned in the cen ButtonCell is enabled, but has no icon, tag, target, action, or key equivalent associated with it. The bordered, and is of type NX_MOMENTARYPUSH.

initIconCell:, initTextCell:

initIconCell:(const char *)iconName

Initializes and returns the receiver, a new ButtonCell instance that displays an icon. iconName is t NXImage that will be used for the Button's icon. The new ButtonCell is enabled, bordered, and is NX_MOMENTARYPUSH.

This is the designated initializer for ButtonCells that display icons.

initTextCell:, init

initTextCell:(const char *)aString

Initializes and returns the receiver, a new ButtonCell instance that displays a title. aString is the tit will be displayed in the user's default system font (as set with the Preferences application), 12.0 po the center. The new ButtonCell is enabled, is bordered, and is of type NX_MOMENTARYPUSH.

This is the designated initializer for ButtonCells that display titles.

initIconCell:, init

(BOOL)isBordered

Returns YES if the ButtonCell has a border, NO if not. A ButtonCell's border isn't the single line borders instead, it's a raised bezel (ªbezelº usually refers to a depressed bezel, as seen on FormCel

setBordered:

(BOOL)isOpaque

Returns YES if the ButtonCell draws over every pixel in its frame, NO if not. The ButtonCell is o transparent and if it has a border.

isBordered, setBordered:, isTransparent, setTransparent:

(BOOL)isTransparent

Returns YES if the ButtonCell is transparent, NO if not. A transparent ButtonCell never draws an receive mouse-down events and track the mouse properly.

setTransparent:, isOpaque

(unsigned short)keyEquivalent

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

setKeyEquivalent:, setKeyEquivalentFont:, setKeyEquivalentFont:size:

performClick:sender

If this ButtonCell is contained in a Control, then invoking this method causes the ButtonCell to act clicked it.

read:(NXTypedStream *)stream

Reads the ButtonCell from the typed stream stream. Returns self.

write:

setAltIcon:(const char *)iconName

Sets the ButtonCell's alternate icon by name iconName is the name of the NXImage to be displaye redrawn if possible, and returns self.

A ButtonCell's alternate icon is only displayed if the ButtonCell highlights or shows its alternate s contents.

contents.

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

setAltTitle:(const char *)aString

Sets the title that the ButtonCell displays in its alternate state to aString. Doesn't display the Butto
autodisplay is on in the ButtonCell's View. Returns self.

The alternate title is shown only if the ButtonCell changes its contents when highlighting or displa

altTitle:, setTitle:, setType:

setBordered:(BOOL)flag

If flag is YES, the ButtonCell displays a border if NO, the If flag is YES, the ButtonCell displays a
ButtonCell doesn't display a border. A ButtonCell's border is not the single line or most other Cel
a raised bezel (ªbezelº usually refers to a depressed bezel, as seen on FormCells, for example). Yo
setBezeled: with a ButtonCell. This method redraws the ButtonCell if the bordered state changes.

isBordered

setDoubleValue:(double)aDouble

If aDouble is 0.0, sets the ButtonCell's state to 0 (the normal state) if aDouble is nonzero, sets it to
Returns self.

doubleValue, setFloatValue:, setIntValue:, setStringValue:

setFloatValue:(float)aFloat

If aDouble is 0.0, sets the ButtonCell's state to 0 (the normal state) if aDouble is nonzero, sets it to
Returns self.

floatValue, setDoubleValue:, setIntValue:, setStringValue:

setFont:fontObject

Sets the Font used to displaying the title and alternate title. Does nothing if the cell has no title or a
self.

If the ButtonCell has a key equivalent, its Font is not changed, but the key equivalent's Font size is
new title Font.

setKeyEquivalentFont:, setKeyEquivalentFont:size:

NX_CHANGEGRAY: The ButtonCell swaps the light gray and white pixels on the its backgro

NX_CHANGEBACKGROUND: Same as NX_CHANGEGRAY, but only background pixels

If both NX_CHANGEGRAY and NX_CHANGEBACKGROUND are specified, both are recorde used depends on the ButtonCell's icon. If there is no icon, or if the icon has no alpha (tranparency NX_CHANGEGRAY is used. If the icon does have alpha data, NX_CHANGEBACKGROUND i gray/white swap of the background to show through the icon's transparent pixels.

highlightsBy, setShowsStateBy:, showsStateBy

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.

A ButtonCell's icon is displayed when the ButtonCell is in its normal state, or always if the Button or show state by changing its contents.

setIcon:position:, icon, setIconPosition:, setImage:, setAltIcon:, setAltImage:, + findImageNam setType:

setIconPosition:(int)aPosition

Sets the position of the icon when a ButtonCell simultaneously displays both text and an icon. aPo 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, setAlignment: (ActionCell)

setImage:image

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

A ButtonCell's NXImage is displayed when the ButtonCell is in its normal state, or all the time for doesn't change its contents when highlighting or displaying its alternate state.

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

setIntValue:(int)anInt

Sets the ButtonCell's state to 1 if anInt is nonzero, 0 otherwise. Returns self.

alternate image to nil before using this method.

keyEquivalent, setKeyEquivalentFont:, setKeyEquivalentFont:size:, performKeyEquivalent: (Bu

setKeyEquivalentFont:fontObject

Sets the Font used to draw the key equivalent, and has the ButtonCell redrawn if possible. Does no already an icon associated with this ButtonCell. The default Font is the same as that used to draw

setKeyEquivalentFont:size:

setKeyEquivalentFont:(const char *)fontName size:(float)fontSize

Sets by name and size the font used to draw the key equivalent, and has the ButtonCell redrawn if if there is already an icon associated with this ButtonCell. The default Font is the same as that use Returns self.

setKeyEquivalentFont:

setParameter:(int)aParameter to:(int)value

Sets the value of one of a number of frequently accessed flags for a ButtonCell to value, and return normally need to use this method since all of these flags can be set through specific methods (for e setHighlightsBy:, and so on). The following table lists each constant used to identify a parameter setting and retrieving the value for that parameter:

getParameter:, setKeyEquivalent:

setShowsStateBy:(int)aType

Sets the way the ButtonCell indicates its alternate state. aType should be the logical OR of one or constants:

NX_NONE (the default): The ButtonCell doesn't change. This flag is ignored if any others ar

NX_CONTENTS: The ButtonCell displays its alternate icon and/or title.

NX_CHANGEGRAY: The ButtonCell swaps the light gray and white pixels on its backgroun

NX_CHANGEBACKGROUND: Same as NX_CHANGEGRAY, but only the background pix

If both NX_CHANGEGRAY and NX_CHANGEBACKGROUND are specified, both are recorde behavior depends on the ButtonCell's icon. If there is no icon, or if the icon has no alpha (tranpare NX_CHANGEGRAY is used. If the icon exists and has alpha data, NX_CHANGEBACKGROU the gray/white swap of the background to show through the icon's transparent pixels.

showsStateBy, setHighlightsBy:, highlightsBy

setSound:aSound

Sets the Sound that will be played when the mouse goes down in the ButtonCell, and whenever the ButtonCell while tracking. Be sure to link against the Sound Kit if you use a Sound object. Return

sound

setStringValue:(const char *)aString

Sets the ButtonCell's state to 1 if aString is non-null (even if the string is empty), 0 otherwise. Ret

setStringValueNoCopy:, stringValue

setStringValueNoCopy:(const char *)aString

Sets the ButtonCell's state to 1 if aString is non-null (even if the string is empty), 0 otherwise. Ret

setStringValue:, stringValue, setDoubleValue:, setFloatValue:, setIntValue:

setTitle:(const char *)aString

Sets the title displayed by the ButtonCell when in its normal state to aString. This title is always sh that don't use their alternate contents when highighting or displaying their alternate state. Redraws and returns self.

setTitleNoCopy:, title, setAltTitle:

A transparent ButtonCell never draws, but does track the mouse and send its action normally. A tr
useful for sensitizing an area on the screen so that an action gets sent to a target when the area rece

isTransparent, isOpaque

setType:(int)aType

Sets the way the ButtonCell highlights while pressed, and how it shows its state. Redraws the But
returns self. aType can be one of the following constants (as described in the Button class specific
method description):

NX_MOMENTARYPUSH
NX_MOMENTARYCHANGE
NX_PUSHONPUSHOFF
NX_ONOFF
NX_TOGGLE
NX_SWITCH
NX_RADIOBUTTON

setType: (Button), setHighlightsBy:, setShowsStateBy:

(int)showsStateBy

Returns the logical OR of flags that indicate the way the ButtonCell shows its alternate state. See s
the list of flags.

setShowsStateBy:, highlightsBy, setHighlightsBy:

sound

Returns the Sound played when the ButtonCell gets a mouse-down event, and whenever the cursor
ButtonCell while tracking.

setSound:

(const char *)stringValue

Returns ^ao (an empty string) if the ButtonCell's state is 1 (the alternate state), or NULL if the state

setStringValue:, setStringValueNoCopy:, doubleValue, floatValue, intValue

(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:, setTitleNoCopy:

trackMouse:inRect:ofView: (Cell)

write:(NXTypedStream *)stream

Writes the receiving ButtonCell to the typed stream stream. Returns self.

read: