BNCS Generic Control

Qt Custom Control

Written by Dave Yates

Description

This is an all-round control that can be used for most of the hard work of panel building. This control draws together a number of the features that are available in a number of other controls that have evolved over time.

This control can be used in Qt Designer directly but only the default presentation will appear. The full editing capabilities of this control are only available in the BNCS Visual Designer.

This control is “reskinable” in that the code required to draw it is separate from the button itself or it’s settings.

Wherever possible don’t explicitly set parameters – use the default or a stylesheet.

Settings

You can pass any of the following settings to this control in the format:

<parameter>=<value>

Default values (those when a new button is created or when a reset command issued) are shown in **bold**.

Parameter	**Value**	**Notes**
reset		Used to reset all the parameters of this control back to their default state

return	any of the button settings	Return a named parameter from this control
		e.g. send this to the control
		`return=text`
		will return a value of the text setting in the format
		`text=This is the text`

colour.text
colour.background
colour.led
	Any valid Qt colour or nothing.	Set colour of the specified parameter.
		e.g. the following two examples are equivalent:
		`colour.led=red`
		`colour.led=#ff0000`
		Setting the value to zero length string (see example) sets the colour back to the default colour specified in the skin. This is the **default** for these settings.
		e.g.
		`colour.background=`
		`colour.text=`

style	label **button**	The fundamental style of this control – the major differences are the width of the border and whether the control "depresses" when clicked
labelstyle	none plain **lowered** raised blackbox whitebox	The border for this control when the style is label. e.g. `labelstyle=raised`
notify.pressed notify.released	**false** **true**	Determines whether this control notifies on mouse press/release or on mouse enter/leave events.
notify.lostfocus	**true**	Determines whether the release event is used or discarded when it occurs outside the area of the button pressed. i.e. if you press a button, slide your finger off it then release do you get the notification?
doublepush	**false** true	Whether this control requires two presses before emitting notification. With this feature enabled a small box appears above the control telling the user that they should press the button again

pixmap	<filename>	Path to the pixmap to display. This can be any valid Qt pixmap e.g. `pixmap=a.png`
text		The text to write on this button. This text will always overlay a pixmap on this button if there is one.
font.face		The current font face. If no font face is specified (the **default**) then the standard font face as defined in the skin is used.

font.size	10
	–2
	(see notes)
		The current font size to use.
		If a font face is specified then this is the absolute point size used.
		If no font face is specified then this value is the offset from the default skin font size.
		e.g. this provides an absolute font/size
		`font.face=arial`
		`font.size=12`
		e.g. this uses the standard font in a slightly smaller size (note the negative font size)
		`font.face=`
		`font.size=-2`

font.style	bold italic underline overline strikeout	The style of the font – this can be a combination of any of the settings (delimited) e.g font.style=bold,italic
textalign	topleft top topright right bottomright bottom bottomleft left **centre**	How to align the text on this button

pixmapalign	(values as for textalign) Default: **top**	How to align the pixmap on this button
led.align	(values as for textalign) Default: **topright**	How to align the pixmap on this button
pixmapstretch	true **false**	Whether the displayed bitmap should be scaled to fit the button. Stretching bitmaps is usually a bad idea – it's slow and the bitmaps usually end up looking rough.
led	true **false**	Whether an "LED" should be displayed.
led.size	**small** medium large	
background	true **false**	Whether this control is a background button and should always appear behind other controls that sit on top of it
width	0	Sets the minimum width for this control (may be resized larger than this value)
height	0	Sets the maximum width for this control (may be resized larger than this value)
sizepolicy	setminimum **asdesigned** shrinktofit	This is what to do if the text/pixmap on this button exceeds the amount of available space to display it. setminimum changes the size of the button to make it big enough for the text/pixmap to fit asdesigned (or null entry) does nothing shrinktofit reduces the size of the text and/or inserts new line characters to make the text fit.

stylesheet		Stylesheet entry to use for this button (the stylesheet entry is an external list of any combination of the above settings)
statesheet		Statesheet entry to use for this button (the statesheet entry is an external list of any combination of the above settings)

Notifications

Notifications are also in the format:

<parameter> = <value>

Parameter	**Value**	**Notes**
button	pressed **released**	The notifications are only made if they are enabled in the settings above. Pressed has to be specifically enabled. Released has to be specifically disabled.

Future Enhancements

Browse

Developers Notes

The browse feature has been removed from this control (as it didn't work properly) but will inevitably make a return.

Version 4.5.16.0 changed the way that this button draws in that the drawing / mouse handling was split out into separate functions. This was to enable reuse of the drawing code in other projects. Most of the functionality of bncs_control has disappeared into this other class and bncs_control code simply makes calls into it.