

Custom Control Library 14

Up / Down Control

Auto Buttons

BNCS Smart Custom Controls
From original controls written by David Yates

Copyright © BBC Resources Ltd. 2000

Copyright © Siemens 2004-2012

Copyright © ATOS IT Solutions & Services Ltd. 2011-2015

Document updated: Steve Jensen 2012-2016

Description

I was starting to get a feeling of deja-vu writing the code behind a range of buttons used on some Axon / Avitel / Vistek kit (PAL -SDI, proc-amps etc.). A lot of the button groups worked very much the same and the ApplCore behind the controls was fiddly - particularly when you know there's a real chance that you're going to be asked "yes its all very good but can I have those buttons as double press?"

This library was created so that it was possible to use common BNCS control groups on panel with no or minimal code behind it. The association between the control and the info-driver slot it controls is made at design time and does not have to be separately coded into the panel. Colours are decided when you place the control on the panel and does not have to be specified later. It all helps to group all these features together in one place.

bbc_cc14.dll is "Smart buttons" that register with the BNCS network independently of the panel application and receive their own revertives and are able to set crosspoints / GPIs / infodriver slots.

Button types

There are 4 controls in this library. 2 versions of an Up / Down control and 2 versions of an Auto Button.

The **Up/Down control** can be used to set the value of a variable parameter, say a video gain value. The two variations of the control give either single up/down buttons or up/down buttons with "big up" and "big down" buttons for larger steps.

The **Auto Buttons** enable you to set the value in an infodriver slot (or the source to a router destination). Nothing unusual in this except that the text to display and the values to use are set on the control when you place it on the panel. E.g. a standards converter may use the values 0 and 1 in an infodriver slot to represent PAL and NTSC respectively. You can tell the control that when it sees a value of 0 illuminate a button you have specified as displaying "PAL" and if it gets a return value of 1 illuminate the "NTSC" button.

Where you can't use *bbc_cc14.dll* buttons

There are some drivers where the command and reply slots are not the same. i.e. to change the input from PAL to NTSC you write 0 and 1 to slot 101 but the reply is returned in slot 102 (this sort of way of working applies to some early BNCS drivers). These controls are not suitable for this sort of application.

CSI Shim *bbc_smcc.exe* (Not used in V3)

This is an application that sits between the smart buttons and CSI providing a "registration fan-out". The problem with Smart buttons is that they need to register individually with CSI. CSI has only got a limited number of clients that it can support and would soon run out when using smart controls. The *bbc_smcc.exe* application concentrates the simple smart button registrations efficiently so that all smart buttons appear as one CSI registration. Currently *bbc_smcc.exe* will support up to 256 smart buttons.

Using Smart Buttons as a group

It is all very well setting the parameters of a control at design time but it is very likely that the destination device will change i.e. an equaliser panel will need to be "directed" to any one of several different equalisers.

Let us say that the equalisers are arranged as gain, chroma, black, hue in slots *nn1* - *nn4* where *nn* is the equalizer number, i.e. equaliser 1 is slots 11 - 14, equaliser 22 is slots 221 - 224.

The Index parameter of the 4 smart button used can be set at **design time** to the values 1 – 4 for the 4 parameters.

At **run time** the index offset can be changed to point at the right equaliser.

If the ID of the controls used is 101 – 104 on panel 1 the ApplCore command:

```
CT 'DESTOFFSET=220' 1 101 104
```

will change all 4 controls to point at equaliser 22. Remember that 220 is the **offset** that is added to the design time **index** so the individual controls will point at slots 221 – 224.

If the destination device number also changes this can be done with the following command:

```
CT 'DESTDEVICE=123' 1 101 104
```

All the parameters for a control can be set in this way – see the documentation on the individual components to see a complete list.

Changing the destination of a control does not cause it to poll for the latest value. To do this write use the command:

```
CT 'POLL=NOW' 1 101 104
```

Important note:

Changing the device number / index offset on the fly is OK but it must be noted that each individual control will generate its own Info Poll command. There are problems with this:

- With the cache on and valid data available in the cache CSI will only service 32 simultaneous poll commands (so put 100 bbc_cc14 buttons on a panel and only the first 32 will work).
- With CSI cache off every poll command will hit the network. This does not make efficient use of the available network bandwidth.

The best work around for this feature is for the **panel** to poll the destination device (and so don't use "POLL=NOW") when it changes the button setup. This relies on a feature whereby a response to a poll command is sent to every client that is registered. The sequence of events in an ApplCore panel would be:

```
CT 'DESTOFFSET=220' 1 101 104:EI SET THE OFFSET FOR THE GROUP  
CT 'DESTDEVICE=123' 1 101 104:EI SET THE DEVICE FOR THE GROUP  
IP 123 220 224:EI POLL THE DEVICE TO REQUEST ALL 4 SIMULTANEOUSLY
```

The panel itself doesn't need to be registered with the destination device for this to work

Future Enhancements

This control has been implemented "on the cheap" in that it subclasses *bbc_cc09.dll* (as written by Mr Tim Alden). This means that every control that you see is one custom control and these have been wrapped up by *bbc_cc14.dll* to give the required functionality. This means for a custom control that uses 10 separate buttons you actually create 11 windows (1 window for each button plus the *bbc_cc14* window control. This isn't efficient. Future versions of this library will manage the button drawing itself requiring only one window. This will be significantly more efficient than could possibly be coded in *ApplCore*.

Notes

- This type of control uses the title field at design time to store all of the parameters needed. This means that "Edit as Text" isn't really a lot of use for editing this type of control as you can see for this example:

```
CONTROL "22123000100000714&ARIAL&PAL&0&2F80&NTSC&1&1F80&", 101, "BBC_CC14_Button", 3 |  
WS_CHILD | WS_VISIBLE, 55, 138, 40, 70
```

- This control can be used on version 1 or version 2 devices but can cause minor problems with version 1 devices. On creating a dialog with a number of *BBC_CC14* controls on – all will register and poll the destination device simultaneously. With version 1 devices, if you aren't using a CSI cache (or it is invalid) these controls will all generate IP, RP or GP commands on the network, all in separate packets and there is some danger that some will be lost. With version 2 this is less of a problem as all the IP, RP or GP commands will get buffered up within CSI and be sent out as one packet. If the CSI cache is valid this isn't a problem with either device.
- Beware using Auto Repeat features on version 1 infodriver externals.

Requirements

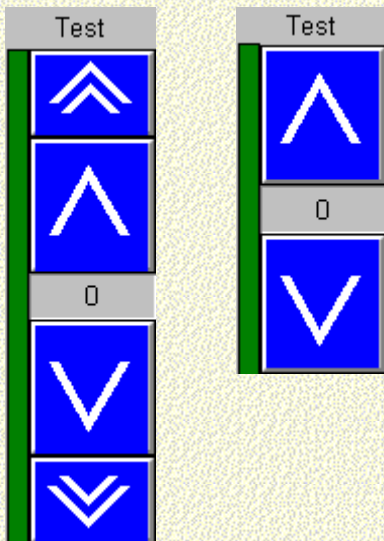
This library requires a current version of *BBC_CC09.DLL* for V2 operation and *BNCS_CC09.DLL* for V3 operation.

Up / Down Control

This control has up down buttons so change the value of either an infodriver slot or a router cross-point. It has the added advantage that it can calculate the value displayed i.e. if the BNCS driver returns a value 0 – 255, this control can be set to display this value as 0 – 100%

There are two versions of this control

1. single up / down controls
2. dual up / down controls where it is desirable to have large step size available.



It is possible to use this control without any ApplCore code whatever by setting the parameters in the design time edit window.

Below is a sample Resource Workshop edit dialog for this type of control.

BBC Multi Up / Down Button Control Style: 1 (CC14)

ID IDC 1 Title Gain Info Drvr	Dest Device <input type="checkbox"/> Router <input checked="" type="checkbox"/> Infodriv <input type="checkbox"/> GPIO Device No 123 Read Index 1 Write Index 0 Offset 0 <input type="checkbox"/> Auto Config If Write Index is 0 then use same slot for read/write	<input checked="" type="checkbox"/> Calculate Level Integer Offset (i) 0 Multiplier (m) 1.000000 Float Offset (f) 0.000000 Decimal Places 0 Units dB Displayed level = $([INPUT + i] * m) + f$ Slot = Index + Offset	<input checked="" type="checkbox"/> OK <input checked="" type="checkbox"/> Cancel <input type="button" value="Run!"/>
Range / Step Top 255 Bottom 0 Step 1 Big Step 5 <input checked="" type="checkbox"/> Auto Repeat			

- The **Range/Step** section specifies the range of the control and how big the steps are. The **Top** parameter refers to the value at the top of the bar graph and the **Bottom** parameter to that value at the bottom of the bar graph. This may seem obvious but in order to change the control to work from the range 0 – 100 to the range 100 to 0 just change over the **Top** and **Bottom** parameters. The **Step** parameters are always positive. The **Big Step** parameter isn't available for single Up / Down controls.
- If one of the buttons is held down and the **Auto Repeat** feature is enabled, the button will send out the next step when it receives a revertive from the previous request. This is implemented in this way so that a button cannot send out parameters faster than the destination device can deal with it. There is a very short delay from pressing and holding a button to it auto-repeating. This is so that the auto repeat function can't interfere with making single step adjustments (in experimentation it was found that with auto repeat on it was possible to accidentally get multiple steps because it wasn't possible to remove your finger fast enough).
- The **Dest Device** section specifies what slot this control is to talk to. These do not have to be set at design time but if they are set the control is completely active within Resource Workshop before even saving.
- For **GPIO** operation the **Top** and **Bottom** should be set to 1 and 0 respectively, with a **Step** and **Big Step** of 1.

BBC Multi Up / Down Button Control Style: 2 (CC14)

ID <input type="text" value="IDC 4"/> Title <input type="text" value="Gain Router 9"/>	Dest Device <input checked="" type="radio"/> Router <input type="radio"/> Infodriv <input type="radio"/> GPIO Device No <input type="text" value="960"/> Read Index <input type="text" value="1"/> Write Index <input type="text" value="0"/> Offset <input type="text" value="0"/> <input type="checkbox"/> Auto Config If Write Index is 0 then use same slot for read/write	<input checked="" type="checkbox"/> Calculate Level Integer Offset (i) <input type="text" value="0"/> Multiplier (m) <input type="text" value="1.000000"/> Float Offset (f) <input type="text" value="0.000000"/> Decimal Places <input type="text" value="0"/> Units <input type="text" value="dB"/> Displayed level = $[(\text{INPUT} + i) * m] + f$ Slot = Index + Offset	<input checked="" type="checkbox"/> OK <input checked="" type="checkbox"/> Cancel
Range / Step Top <input type="text" value="255"/> Bottom <input type="text" value="1"/> Step <input type="text" value="1"/> <input checked="" type="checkbox"/> Auto Repeat			

Setting parameters at run time.

Any of the above parameters may be set at run time by writing a command of the format:

command=value

where *command* and *value* are one of the following parameters

Command	Value
DESTDEVICE	BNCS Device Number
DESTINDEX	Index (destination slot = DESTINDEX+DESTOFFSET) This is equivalent to the READ index
DESTWRITEINDEX	The index to write to (if it is different to the READ slot). Either set this the same as DESTINDEX or leave at 0 (where it is assumed that DESTINDEX = DESTWRITEINDEX)
DESTOFFSET	Index Offset (destination slot = DESTINDEX+DESTOFFSET)
BOTTOM (MIN)	Minimum value – this control will only write values >= this value.
TOP (MAX)	Maximum value – this control will only write values <= this value.
STEP	Single step value
BIGSTEP	Big Step value – must be bigger than STEP and is not used on the single up/down control.
TITLE	Title of the control
UINTS	What units to display in CALCULATELEVEL mode
IOFFSET	Integer offset used in CALCULATELEVEL mode
FOFFSET	Floating point offset used in CALCULATELEVEL mode
MULTIPLIER	Multiplier used in CALCULATELEVEL mode
DECIMALPLACES	Number of decimal places in CALCULATELEVEL mode
CALCULATELEVEL	Whether to display the level directly from the revertive or use the CALCULATELEVEL parameters above.
AUTOREPEAT	On receipt of a revertive whether the control sends another command if the button is still depressed.
DEVICETYPE	The device type: I for info drivers, R for routers or G for GPIO devices.

To force the control to re-poll the device write the command:

POLL=NOW

Calculate Level mode

In this mode the (numeric) return value is used to calculate the displayed value.

The value displayed is:

$$((\text{INPUT} + i) * m) + f$$

where

INPUT is the value returned from the destination device.

i is the integer offset

m is the multiplier

f is the floating point offset

The UNITS parameter is appended to the end of the calculated value.

Return Value

The return value from this type of control is the current numeric value (i.e. the value returned from the infodriver slot).

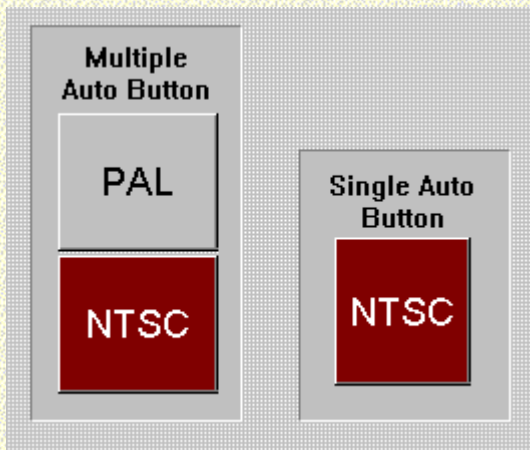
Auto Buttons

There are two versions of this button.

1. A single button that changes caption and colour dependent upon the revertive it receives from the destination device.
2. Multiple buttons that highlight when the destination is in a particular state.

e.g. A device has two states stored in slot 1 of an infodriver – where a slot contents of “0” is PAL, and slot contents of “1” is NTSC.

Below are two working examples of the two types of controls, both of these controls are linked to slot 1 on device 123 and both controls work with the same parameters and values but the results are displayed differently.



Above is the setup dialog as displayed in Resource Workshop for the multiple button (left).

BNCS Multiple Auto Button Style: 3 (CC14)

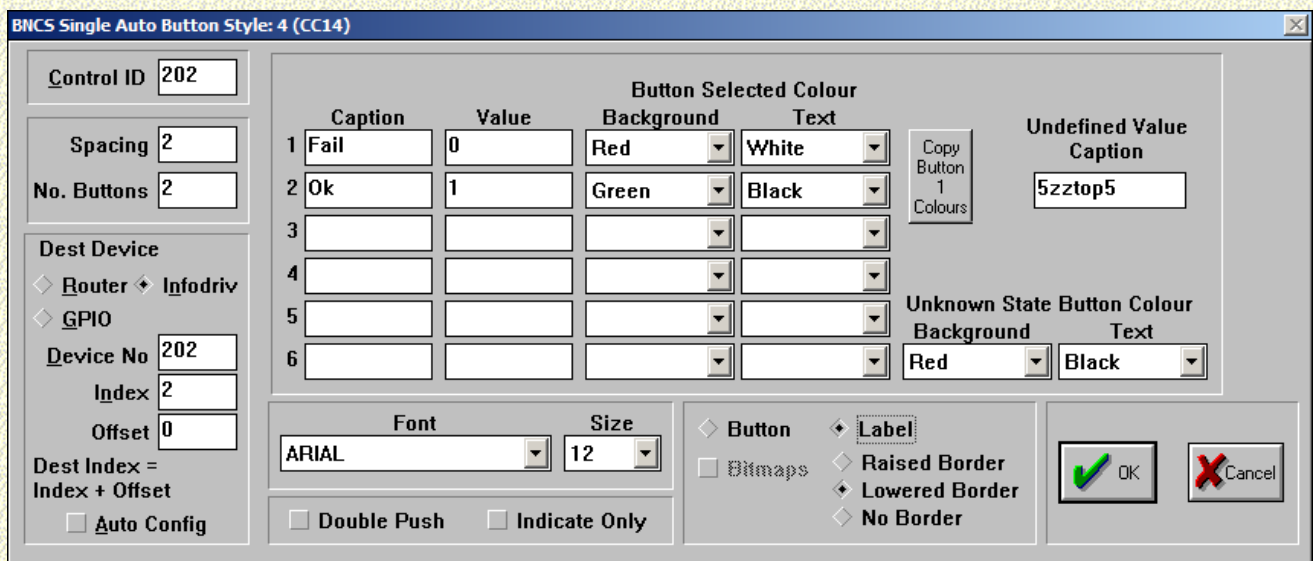
Control ID		Caption		Value	Button Selected Colour		Button Deselected Colour		Copy Button 1 Colours
					Background	Text	Background	Text	
215		1	Freeze	1	Dark Blue	White	Light Grey	Black	
Spacing 2		2	Black	2	Dark Blue	White	Light Grey	Black	
No. Buttons 2		3							
Dest Device		4							
Router Infodriv		5							
GPIO		6							
Device No 0									
Index 1									
Offset 0									
Dest Index = Index + Offset									
<input type="checkbox"/> Auto Config									

Font	Size	Button	Label
ARIAL	12	<input checked="" type="checkbox"/> Raised Border	<input type="checkbox"/> Lowered Border
		<input type="checkbox"/> Double Push	<input type="checkbox"/> Indicate Only
		<input checked="" type="checkbox"/> No Border	

OK Cancel

Taking the sections starting at the top in the first column:

- **Spacing** is the distance between the buttons. This can have a maximum value of 9.
- **No. Buttons** is the number of buttons to display. This value automatically increments when changing the button parameters.
- The **Dest Device** section specifies what slot this control is to talk to. These do not have to be set at design time but if they are set the control is completely active within Resource Workshop before even saving.
- The button parameters section defines the captions that are to be displayed on the buttons and the info driver/GRD/GPIO values that correspond to those captions. For info drivers this can be any string, for GRDs and GPIOs the value must be numeric. In this example there are 2 buttons with captions PAL and NTSC. When the infodriver slot contains the value “0” the PAL button will be highlighted with the **Button Selected Colour**. Other buttons will be displayed with their **Button Deselected Colour**. For ease of editing a “Copy Button 1 colours” has been provided to copy the colours down to all of the controls from the top row.
- The font and size drop down lists determine the font to use for all the buttons.
- The Double push flag is used for important functions where you would require a second push to action the event. When this function is enabled the first push causes the button to go red and the caption changes to “HitAgain”. If the button is hit within the timeout period then the event happens, otherwise it returns to its normal state.
- The Indicate only flag enables this control to be used where you don’t want to be able to change the slot but want to indicate it’s current status.
- The Button / Label flags determine the appearance of the controls.



BNCS Single Auto Button Style: 4 (CC14)

Control ID: 202

Spacing: 2

No. Buttons: 2

Dest Device

Router ☒ Infodriv

GPIO ☒

Device No: 202

Index: 2

Offset: 0

Dest Index = Index + Offset

☐ Auto Config

	Caption	Value	Button Selected Colour Background	Text
1	Fail	0	Red	White
2	Ok	1	Green	Black
3				
4				
5				
6				

Copy Button 1 Colours

Undefined Value Caption: 5zztop5

Unknown State Button Colour Background: Red Text: Black

Font: ARIAL Size: 12

☐ Double Push ☐ Indicate Only

☒ Button ☒ Label

☐ Bitmaps

☒ Raised Border

☒ Lowered Border

☒ No Border

OK Cancel

Below is the similar setup dialog for the **Single Auto Button**.

Many of the parameters are the same but instead the parameters list is a list of *states* rather than buttons. When the infodriver returns a value of “0” the button caption displays PAL and the button changes to the Button Selected Colour. If a parameter other than one of those listed in the “Value” column then the button displays the actual return value in the colour defined by the Unknown State Button Colour.

The only other change is that it is possible to display bitmaps on this control rather than just text.

e.g. in the above example if the Bitmap flag was to be set then the button would display a bitmap called PAL, either from the resource file of the project or in the same directory as the application.

Beginning with V2 bbc_cc14.dll version: 1.06.04 and V3 bnCS_Cc14.dll version: 3.04.06 support has been added for “undefined” button state defined text in both style 3 and style 4 buttons. Text values are defined at design time for each of six values of the slot for the button. If the slot value is not defined, then the “Undefined Value” text will be written to the button face.

Setting parameters at run time.

Any of the above parameters may be set at run time by writing a command of the format:

command=value

where *command* and *value* are one of the following parameters

Command	Value
DESTDEVICE	BNCS Device Number
DESTINDEX	Index (destination slot = DESTINDEX+DESTOFFSET)
DESTOFFSET	Index Offset (destination slot = DESTINDEX+DESTOFFSET)
DEVICETYPE	The device type – either I for info drivers or R for routers or G for GPIO devices.

To force the control to re-poll the device write the command:

POLL=NOW

Router Support

Support for the V3 ApplCore router “mask” parameter was added to both V2 and V3 CC14 and CC21 starting with versions:

BBC_CC14.DLL version: 1.5.0, dated: 30Dec2012
BBC_CC21.DLL version: 1.6.0, dated: 30Dec2012
BNCS_CC14.DLL version: 3.03.05, dated: 04Jan2013
BNCS_CC21.DLL version: 3.04.04, dated: 30Dec2012

The new “mask” parameter support is added via the corresponding library .INI file using files: bbc_cc14.ini and bbc_cc21.ini. The “mask” parameter support is identical for the V2 and V3 libraries as well as for the CC14 and CC21 versions as follows:

```
//  
// bbc_cc14.ini file example  
//  
[RcDevicesMask]  
960='FULL'  
961='All'  
962='Forward'  
963='VISION'  
964='REVFACS'  
  
//  
// bbc_cc21.ini file example  
//  
[RcDevicesMask]  
960='FULL'  
961='All'  
962='Forward'  
963='VISION'  
964='REVFACS'
```

The router mask is specified for a router as in the examples above, i.e. the 960 entry specifies for router 960 the mask is specified as the string value 'FULL'. If a router is not specified via an entry in the .INI file then the mask parameter is not issued.

Example equivalent V3 ApplCore commands:

RC 960 12 34 'FULL'	routes 960 12 and 34 using mask of 'FULL'
RC 900 77 88	routes 900 77 and 88 no mask is specified

Notes

There are only a limited number of characters (94) that can be used for a title within resource workshop. This is an issue for the Auto Buttons where the length of the individual captions used can cause the 94 character limit to be exceeded. In practice for short captions it was found that 6 was the maximum number of buttons that could be supported. It may be that it is not possible to get this number of buttons if the captions or values are longer than a few characters. If using the Multiple Auto Buttons then simply use another group of buttons linked to the same slot. It is not possible to use Single Auto Buttons with a large number of long parameters.

If the number of available characters has been exceeded then on hitting OK closing the edit window you'll get a message telling you it's all gone horribly wrong.

Document Revision History

- Initial Release 2000 – BBC Resources.
- Release 11-March-2005 – Minor formatting changes.
- Updated 22-March-2013 – Added RC ‘mask’ parameter additions. Updated “BNCS” logo. Updated copyright notice. Steve Jensen
- Corrected a number of errors in the text of this document and updated to include support for GPIO.
GPIO support in the libraries begins with released versions: V2 bbc_cc14.dll version:1.6.4, dated: 06-June-2014 and V3 bnCS_cc14.dll version: 3.04.06, dated: 06-June-2014
The destination selector in Borland Workshop now includes InfoDrv, Router and GPIO. Steve Jensen 09-June-2014
- Corrected a number of errors in the text of this document Steve Jensen 24-June-2014
- Added support for 4.5 environment. Bug 2637 BNCS_CC14.dll Version: 3.04.06, Dated: 25-Feb-2015
- Added support for “undefined” button state text in style 3 and style 4 buttons. V2 bbc_cc13.dll version: 1.06.04 and V3 bnCS_cc14.dll version: 3.04.06 and later. July-2015
- 4.5 environment full Win32 path length. Bug 2637. bnCS_cc14.dll version: 3.04.12 and later. August-2015
- Bug fixes to “undefined” button value support. Removed from Style 3 buttons, BBC_CC14.DLL version: 1.06.09, dated: 19May2016