ControlZ 1.2 Custom Control Library
User Manual

C. van Zwynsvoorde.
E-Mail: cvzwynsv@vmprofs.estec.esa.nl

Table of contents

1. Overview
2. Scaler
3. Dial
4. Tuner
5. Combo box
6. List box
7. Static Text
8. Static Link
9. How to register

1. Overview

ControlZ is a custom control library (DLL).
It basically aims at making developers job easier.
It is Borland's Resource WorkShop 1.01+ compliant.

That means you can customize the controls and use them directly with Borland's Resource WorkShop. To do this you need to select Options | Install Control Library when editing a dialog box. If you are not a Borland user, you can still use the ControlZ.DLL at run-time but will probably experience problems if trying to use it interactively in your resource editor. In that case you might want to let me know (see the at end this file).

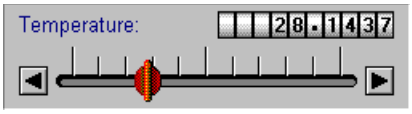
Currently, 7 types of controls have been implemented:
- an analogue scaler,
- an analogue dial,
- an analogue tuner,
- a new type of hiererchic combo-box,
- a new type of hierarchic list-box with horizontal (caption) scrolling,
- an extended static text control,
- an extended arrowed link static control.

ControlZ has been designed with the intention of putting more power in the DLL and less in your application. The controls have indeed a variety of options with default behaviours and even built-in "demo" capabilities. Hence the library can be used for two purposes:
1. providing powerful controls to your application, and
2. making models of your future applications that still appear to do something. You do that only by interactively defining the resources in your resource editor. It makes you save time at this stage were you are often in a rush to get something to show in order to get funds for your future application.

Remember ControlZ.DLL is not a VBX ! Thus it has no complicated tricks to get it working with C and the resource editor and/or compiler. Also this means you may expect it to run on future versions of Windows (which is not sure for VBX controls).

The ControlZ.DLL that comes in the package (usually `controlz.zip`) is fully operational. You will only get some "unregistered copy" reminders if you are not registered. It is associated with a header file called ControlZ.H and the on-line documentation contained in the ControlZ.HLP and ControlZ.WRI files. The WRI file has the same contents as the HLP file but can be more easily printed.

2. Scaler



Class Name:
The class name is "CZScaler"

Sizing:
When resizing, the control will try to keep the things at their best place. That is:

1. The scaler groove will occupy all the available horizontal space.
2. The groove will be kept at the bottom.
3. The counter will be placed at the top-right corner
4. The caption will occupy the remaining place in the top-left corner (possibly being wrapped on several lines).

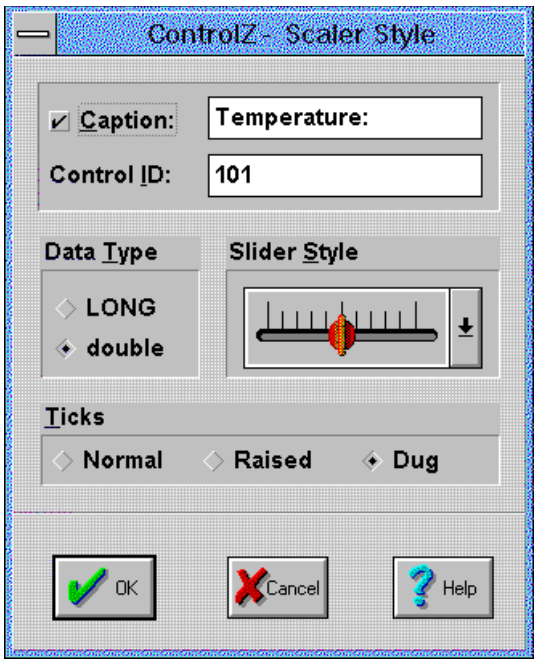
You should be aware that making the control too small will result in some overlapping.

Font:
The control uses the same font as the parent window except it is not bold. In order to get this effect, you must instruct the parent window (usually the dialog box) to use a font that supports this selection, which the default font (Fixedsys) does not. Borland suggests you design your dialog boxes with the Arial 8 font.

The caption text will appear in blue.

In the current version, there is no way to change the text color and/or font. The `WM_SETFONT` message will have no effect. Still `WM_GETFONT` works fine.

Styles:



The scaler control supports the following styles, which can also be selected from the style dialog box shown above:

`CZSS_CAPTION` or `CZSS_NOCAPTION`: indicates whether the control consists only of the scaler with no counter and caption. Note: if you want no caption but still the counter, just make the caption string empty.

`CZSS_TYPELONG` or `CZSS_TYPEDOUBLE`: indicates whether the scaler will use variables of type (long) or (double) in the C coding.

`CZSS_SLIDER_0` to `CZSS_SLIDER_9`: indicates the kind of slider that should be used.

`CZSS_TICKS_NORMAL`, `CZSS_TICKS_RAISED` or `CZSS_DUG`: indicates the style used for the graduations.

Messages:

`WM_SETFONT`:
Will have no effect.

`WM_GETFONT` and `WM_SETTEXT`:
As expected.

`CZM_SETRANGEMIN` and `CZM_SETRANGEMAX`:
Used at any time to set the minimum and maximum values allowed for the counter. If this results in the current position to be outside the range, then the position will be corrected too. You can't set a min value of more that the

current max value. You can't set a max value of less than the current min value.
wParam = NULL
(LONG)lParam = min or max value if CZSS_TYPELONG.
*((double FAR *)lParam) = min or max value if CZSS_TYPEDOUBLE.
return value = NULL

CZM_GETRANGEMIN and CZM_GETRANGEMAX:
Used at any time to retrieve the minimum and maximum valued allowed for the counter.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZSS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZSS_TYPEDOUBLE.
return value = min or max value if CZSS_TYPELONG.
return value = NULL if CZSS_TYPEDOUBLE.

CZM_SETINC:
Used at any time to set the increment, that is the value that will be added to or subtracted from the current positions when a button is pressed. Negative and positive values are equivalent: only the absolute value is taken into account.
wParam = NULL
(LONG)lParam = increment value if CZSS_TYPELONG.
*((double FAR *)lParam) = increment value if CZSS_TYPEDOUBLE.
return value = NULL

CZM_GETINC:
Used at any time to retrieve the value of the increment.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZSS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZSS_TYPEDOUBLE.
return value = increment value if CZSS_TYPELONG.
return value = NULL if CZSS_TYPEDOUBLE.

CZM_SETPOS:
Used at any time to set the position of the counter (and the slider). This is relative to the specified range (see before). Values outside the allowed range are truncated to either the minimum or maximum acceptable value.
wParam = NULL
(LONG)lParam = position value if CZSS_TYPELONG.
*((double FAR *)lParam) = position value if CZSS_TYPEDOUBLE.
return value = NULL

CZM_GETPOS:
Used at any time to retrieve the current position of the counter.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZSS_TYPELONG.

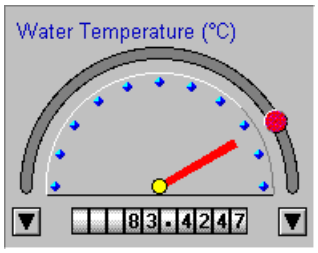
(double FAR *)lParam = pointer to the returned value if CZSS_TYPEDOUBLE.
return value = increment value if CZSS_TYPELONG.
return value = NULL if CZSS_TYPEDOUBLE.

CZM_INCPOS:
Used at any time to either increase or decrease the current position by an amount called the increment (see CZM_SETINC). If this would result in the position being outside of the allowed range, then the position will be truncated to either the minimum or maximum acceptable value. Sending this message is equivalent to pressing the buttons at the left and right of the control. As far as the CPU allows it, the buttons generate 20 CZM_INCPOS messages per second.
wParam = TRUE for increment. FALSE for decrement.
lParam = NULL
return value = NULL

Notification Messages:
CZN_POSCHANGE:
Sent back to the parent window (usually the dialog box) whenever the position has changed.
iMessage = WM_COMMAND
wParam = control window handle
lParam = MAKELONG(control id, CZN_POSCHANGE)
return value = not used

Default Behaviour:
The default style is CZSS_CAPTION | CZSS_TYPELONG | CZSS_SLIDER_0 | CZSS_TICKS_DUG.
The default caption is "record:".
The default range is 0 to 100.
The default increment is 1.

3. Dial



Class Name:
The class name is "CZDial"

Sizing:
When resizing, the control will try to keep the things at their best place. That is:

1. The counter will be horizontally centred, at the bottom.
2. The caption (if any) will be at the top, possibly occupying several lines.
3. The dial will try to occupy the remaining place. Still it will be kept circular, bottom aligned and horizontally centred.

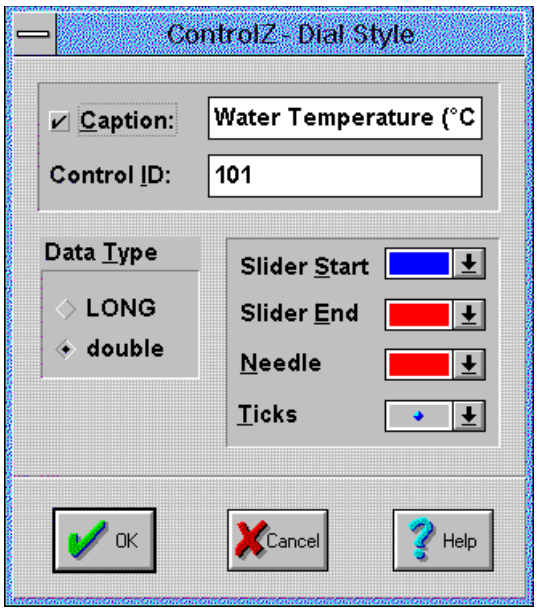
You should be aware that making the control too small will result in some overlapping.

Font:
The control uses the same font as the parent window except it is not bold. In order to get this effect, you must instruct the parent window (usually the dialog box) to use a font that supports this selection, which the default font (Fixedsys) does not. Borland suggests you design your dialog boxes with the Arial 8 font.

The caption text will appear in blue.

In the current version, there is no way to change the text color and/or font. The `WM_SETFONT` message will have no effect. Still `WM_GETFONT` works fine.

Styles:



The dial control supports the following styles, which can also be selected from the style dialog box shown above:

`CZDS_CAPTION` or `CZDS_NOCAPTION`: indicates whether the control consists only of the dial with no counter and caption. Note: if you want no caption but still the counter, just make the caption string empty.

`CZDS_TYPELONG` or `CZDS_TYPEDOUBLE`: indicates whether the dial will use variables of type (long) or (double) in the C coding.

`CZDS_SLIDERSTART_0` to `CZDS_SLIDERSTART_15`: specifies the color to be used for the slider when it is at the minimum position.

`CZDS_SLIDEREND_0` to `CZDS_SLIDEREND_15`: specifies the color to be used for the slider when it is at the maximum position.

For each position between the minimum and maximum, the slider will be affected a color derived from those two end-colors by a linear extrapolation. That allows you to animated the slider for example from blue to red (cold to hot!). If you don't want it to be animated, just specify the same color for both ends.

In ControlZ, 16 predefined colors are used, coded from 0 to 15. To minimize differences that might occur when moving from one hardware configuration to the other, we don't rely on the system colors but define a proprietary 16-colors palette out of the most commonly used colors. Still it is most likely to just correspond with your system colors. Colors are coded as follows:

0 = black	1 = dark blue	2 = dark green	3 = dark red
4 = dark yellow	5 = dark pink	6 = medium blue	7 = light gray

8 = dark gray	9 = blue	10= green	11 = red
12 = yellow	13 = pink	14 = light blue	15 = white.

CZDS_NEEDLE_0 to CZDS_NEEDLE_15: specifies the needle color.

CZDS_TICKS_0 to CZDS_TICKS_3: indicates the style used for the graduation marks. There are four of them, figuring colored, raised or dug small pyramids.

Messages:

WM_SETFONT:
Will have no effect.

WM_GETFONT and WM_SETTEXT:
As expected.

CZM_SETRANGEMIN and CZM_SETRANGEMAX:
Used at any time to set the minimum and maximum values allowed for the counter. If this results in the current position to be outside the range, then the position will be corrected too. You can't set a min value of more that the current max value. You can't set a max value of less than the current min value.
wParam = NULL
(LONG)lParam = min or max value if CZDS_TYPELONG.
*((double FAR *)lParam) = min or max value if CZDS_TYPEDOUBLE.
return value = NULL

CZM_GETRANGEMIN and CZM_GETRANGEMAX:
Used at any time to retrieve the minimum and maximum valued allowed for the counter.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZDS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZDS_TYPEDOUBLE.
return value = min or max value if CZDS_TYPELONG.
return value = NULL if CZDS_TYPEDOUBLE.

CZM_SETINC:
Used at any time to set the increment, that is the value that will be added to or subtracted from the current positions when a button is pressed. Negative and positive values are equivalent: only the absolute value is taken into account.
wParam = NULL
(LONG)lParam = increment value if CZDS_TYPELONG.
*((double FAR *)lParam) = increment value if CZDS_TYPEDOUBLE.
return value = NULL

CZM_GETINC:
Used at any time to retrieve the value of the increment.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZDS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZDS_TYPEDOUBLE.
return value = increment value if CZDS_TYPELONG.
return value = NULL if CZDS_TYPEDOUBLE.

CZM_SETPOS:
Used at any time to set the position of the counter (and the slider). This is relative to the specified range (see before). Values outside the allowed range are truncated to either the minimum or maximum acceptable value.
wParam = NULL
(LONG)lParam = position value if CZDS_TYPELONG.
*((double FAR *)lParam) = position value if CZDS_TYPEDOUBLE.
return value = NULL

CZM_GETPOS:
Used at any time to retrieve the current position of the counter.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZDS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZDS_TYPEDOUBLE.
return value = increment value if CZDS_TYPELONG.
return value = NULL if CZDS_TYPEDOUBLE.

CZM_INCPOS:
Used at any time to either increase or decrease the current position by an amount called the increment (see CZM_SETINC). If this would result in the position being outside of the allowed range, then the position will be truncated to either the minimum or maximum acceptable value. Sending this message is equivalent to pressing the buttons at the left and right of the control. As far as the CPU allows it, the buttons generate 20 CZM_INCPOS messages per second.
wParam = TRUE for increment. FALSE for decrement.
lParam = NULL
return value = NULL

Notification Messages:
CZN_POSCHANGE:
Sent back to the parent window (usually the dialog box) whenever the position has changed.
iMessage = WM_COMMAND
wParam = control window handle
lParam = MAKELONG(control id, CZN_POSCHANGE)
return value = not used

Default Behaviour:

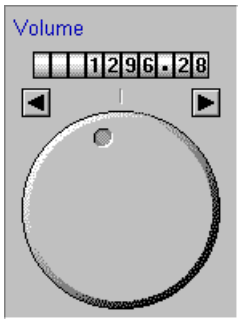
The default style is `CZDS_CAPTION | CZDS_TYPELONG | CZDS_SLIDERSTART_9 | CZDS_SLIDEREND_11 | CZDS_NEEDLE_11 | CZDS_TICKS_0`.

The default caption is `"Speed:"`.

The default range is 0 to 100.

The default increment is 1.

4. Tuner



Class Name:
The class name is "CZTuner"

Sizing:
When resizing, the control will try to keep the things at their best place. That is:
1. The caption (if any) will be at the top, possibly occupying several lines.
2. The two buttons will be placed on above the wheel, and aligned horizontally with it.
3. The counter (if any) will be placed above the buttons and horizontally centred.
4. The will try to occupy the remaining place, still being centred horizontally and bottom aligned.

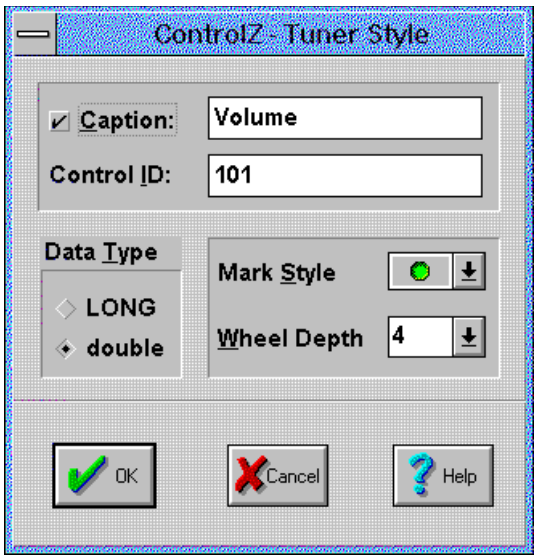
You should be aware that making the control too small will result in some overlapping.

Font:
The control uses the same font as the parent window except it is not bold. In order to get this effect, you must instruct the parent window (usually the dialog box) to use a font that supports this selection, which the default font (Fixedsys) does not. Borland suggests you design your dialog boxes with the Arial 8 font.

The caption text will appear in blue.

In the current version, there is no way to change the text color and/or font. The `WM_SETFONT` message will have no effect. Still `WM_GETFONT` works fine.

Styles:



The tuner control supports the following styles, which can also be selected from the style dialog box shown above:

`CZTS_CAPTION` or `CZTS_NOCAPTION`: indicates whether the control consists only of the dial with no counter and caption. Note: if you want no caption but still the counter, just make the caption string empty.

`CZTS_TYPELONG` or `CZTS_TYPEDOUBLE`: indicates whether the dial will use variables of type (long) or (double) in the C coding.

`CZTS_WHEELDEPTH_0` to `CZTS_WHEELDEPTH_15`: specifies the depth of the wheel, in pixel units. Allowed depths are 0 to 15.

`CZTS_MARK_0` to `CZTS_MARK_15`: specifies the style of the mark. That is the thing you drag to make the wheel turn. Sixteen styles are predefined, figuring big or small holes or raised spots, in gray, red, green or blue.

Messages:

`WM_SETFONT`:
Will have no effect.

`WM_GETFONT` and `WM_SETTEXT`:
As expected.

`CZM_SETTURN`:
Used at any time to set the range covered by one turn of the wheel. Negative and positive values are equivalent: only the absolute value is taken into account. If this results in the current position to be outside the range, then the position will be corrected too.
At the beginning, the counter has value 0. The 0 position always corresponds

to having the mark on the top, in front of the small vertical tick. The tuner control offers you at the same time precise control on the counter value, and an unlimited range of values. This is a unique capability and a big improvement on any type of scaler.
wParam = NULL
(LONG)lParam = turn range if CZTS_TYPELONG.
*((double FAR *)lParam) = turn range if CZTS_TYPEDOUBLE.
return value = NULL

CZM_GETTURN:
Used at any time to retrieve the range covered by one turn of the wheel.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZTS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZTS_TYPEDOUBLE.
return value = min or max value if CZTS_TYPELONG.
return value = NULL if CZTS_TYPEDOUBLE.

CZM_SETINC:
Used at any time to set the increment, that is the value that will be added to or subtracted from the current positions when a button is pressed. Negative and positive values are equivalent: only the absolute value is taken into account.
wParam = NULL
(LONG)lParam = increment value if CZTS_TYPELONG.
*((double FAR *)lParam) = increment value if CZTS_TYPEDOUBLE.
return value = NULL

CZM_GETINC:
Used at any time to retrieve the value of the increment.
wParam = NULL
(LONG FAR *)lParam = pointer to the returned value if CZTS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZTS_TYPEDOUBLE.
return value = increment value if CZTS_TYPELONG.
return value = NULL if CZTS_TYPEDOUBLE.

CZM_SETPOS:
Used at any time to set the position of the counter (and the slider). This is relative to the specified range (see before). Values outside the allowed range are truncated to either the minimum or maximum acceptable value.
wParam = NULL
(LONG)lParam = position value if CZTS_TYPELONG.
*((double FAR *)lParam) = position value if CZTS_TYPEDOUBLE.
return value = NULL

CZM_GETPOS:
Used at any time to retrieve the current position of the counter.
wParam = NULL

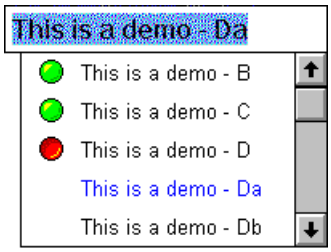
(LONG FAR *)lParam = pointer to the returned value if CZTS_TYPELONG.
(double FAR *)lParam = pointer to the returned value if CZTS_TYPEDOUBLE.
return value = increment value if CZTS_TYPELONG.
return value = NULL if CZTS_TYPEDOUBLE.

CZM_INCPOS:
Used at any time to either increase or decrease the current position by an amount called the increment (see CZM_SETINC). If this would result in the position being outside of the allowed range, then the position will be truncated to either the minimum or maximum acceptable value. Sending this message is equivalent to pressing the buttons at the left and right of the control. As far as the CPU allows it, the buttons generate 20 CZM_INCPOS messages per second.
wParam = TRUE for increment. FALSE for decrement.
lParam = NULL
return value = NULL

Notification Messages:
CZN_POSCHANGE:
Sent back to the parent window (usually the dialog box) whenever the position has changed.
iMessage = WM_COMMAND
wParam = control window handle
lParam = MAKELONG(control id, CZN_POSCHANGE)
return value = not used

Default Behaviour:
The default style is CZTS_CAPTION | CZTS_TYPELONG | CZTS_WHEELDEPTH_4 | CZTS_MARK_1.
The default caption is "Speed:".
The default range is 0 to 100.
The default increment is 1.

5. Combo box



Class Name:
The class name is "CZCombo"

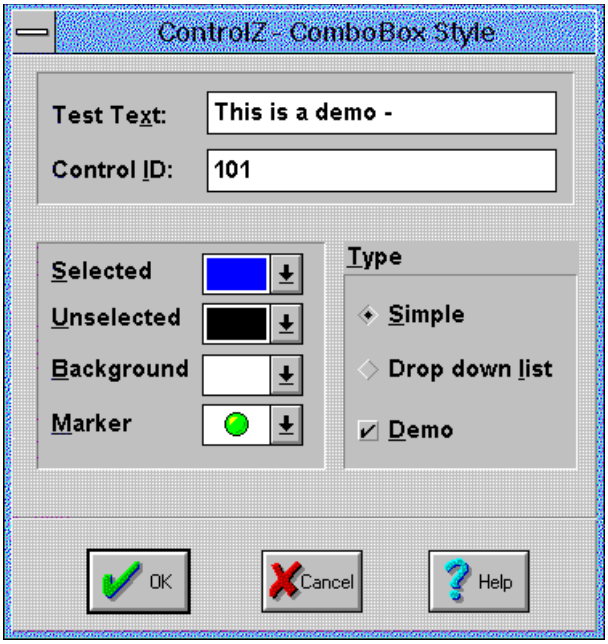
Sizing:
The default combo-box resizing behaviour has been maintained, with the addition that a margin has been reserved on the left for the marker. The marker is the symbol that indicates the presence of section headers.

Font:
The control uses the same font as the parent window except it is not bold. In order to get this effect, you must instruct the parent window (usually the dialog box) to use a font that supports this selection, which the default font (Fixedsys) does not. Borland suggests you design your dialog boxes with the Arial 8 font.

In the current version, there is no way to change the font. The `WM_SETFONT` message will have no effect. Still `WM_GETFONT` works fine.

The colors (background, foreground selected and unselected) can be configured. See the styles section.

Styles:



The control's caption (that is the text that is set by the SetWindowText function), is used as the demo text if the demo checkbox (`CZCBS_DEMO`) is selected. Otherwise it is not used.

The combo-box control supports the following styles, which can also be selected from the style dialog box shown above:

`CZCBS_SELECTED_0` to `CZCBS_SELECTED_15`: specifies the color for the item strings when selected.

In ControlZ, 16 predefined colors are used, coded from 0 to 15. To minimize differences that might occur when moving from one hardware configuration to the other, we don't rely on the system colors but define a proprietary 16-colors palette out of the most commonly used colors. Still it is most likely to just correspond with your system colors. Colors are coded as follows:

0 = black	1 = dark blue	2 = dark green	3 = dark red
4 = dark yellow	5 = dark pink	6 = medium blue	7 = light gray
8 = dark gray	9 = blue	10= green	11 = red
12 = yellow	13 = pink	14 = light blue	15 = white.

`CZCBS_UNSELECTED_0` to `CZCBS_UNSELECTED_15`: specifies the color for the item strings when not selected.

`CZCBS_BACKGROUND_0` to `CZCBS_BACKGROUND_15`: specifies the color for the combo-box background.

`CZCBS_MARKER_0` to `CZCBS_MARK_3`: specifies the style of the marker. That is the symbol placed in the left margin. It is used to indicate that the item is

either a section header or a normal item. When you select a section header, the previously expanded section will collapse and the selected section will automatically expand to show its dependant items. The marker symbol is animated to indicate whether a section is currently expanded or not.

CZCBS_SIMPLE or CZCBS_DROPDOWNLIST: specifies whether the combo-box should comply to the standard CBS_SIMPLE or CBS_DROPDOWNLIST style. Note that the CBS_DROPDOWN standard style has been abandoned because its priciple is quite contrary to having an expanding sections mechanism.

CZCBS_DEMO or CZCBS_NODEMO: specifies whether the control should act as a demo one. This feature has been added to allow you to design dialog boxes, run them and already have controls that "do something". That means that, while being in a preliminary study phase and having programmed nothing yet, you can show your boss or your customer more that a standard empty combo-box.

The demo acting consists of generating 26 sections, each expanding to 26 items. This is done by appending letters ('A' to 'Z', then 'a' to 'z') to the control's caption.

By the time you want to use the control in your real application, you will have to remove the CZCBS_DEMO style. An alternative would be to delete all the demo sections after the control's creation (use the CZM_DELETESECTION message).

Messages:

WM_SETFONT:
Will have no effect.

WM_GETFONT and WM_SETTEXT:
As expected.

CZM_ADDSECTION:
Used at any time to add a section header to the combo-box. The section header will be appended at the end of the list.
wParam = a unique identifier for the section. You are responsible for providing this and checking uniqueness. Non uniqueness is not expected to be fatal but is not "documented". Allowed numbers range from 0 to 65535.
(LPSTR)lParam = section header string.
return value = index of the added string in the list or CB_ERR upon error.

CZM_DELETESECTION:
Used at any time to delete a section, regardless of whether it is currently expanded or not.

wParam = section identifier as specified in the CZM_ADDSECTION message.
lParam = NULL.
return value = number of remaining strings in the combo-box, or CB_ERR upon error. Note that the section identifier not being found is not considered as an error.

CZM_ISSECTION:
Used at any time to determine whether a given string in the list is a section header or a normal item.
wParam = index of the string in the list.
lParam = NULL.
return value = CB_ERR if an error occurs. Non zero if the wParam'th list item is a section header. Zero otherwise.

CZM_FILLSECTION:
Used when a CZN_FILLSECTION notification message is received. This is when a section is about to be expanded and the control needs you to supply the items associated with it. For that purpose you must send CZM_FILLSECTION messages back to the control, for each item that belongs to the section. Note that, when handling the CZN_FILLSECTION notification message, you should not send messages other than CZM_FILLSECTION. The section is considered to be filled when you return from the CZN_FILLSECTION notification message. Any attempt to send CZM_FILLSECTION messages outside the handling of CZN_FILLSECTION will have not effect.
wParam = NULL.
(LPSTR)lParam = item's string.
return value = index of the added string in the list, or CB_ERR upon error.

CB_ADDSTRING and CB_INSERTSTRING:
Provided for compatibility with the standard combo-box. You actually can disable the section mechanism just by adding no section and using those messages instead. This will make the control to act as a standard combo-box but will still provide you with the formatting capabilities (colors, etc.).
Using these two messages when you have sections in the combo box, may produce unpredictable results.

CB_FINDSTRING, CB_GETCOUNT, CB_GETCURSEL,
CB_GETDROPPEDCONTROLRECT, CB_GETDROPPEDSTATE,
CB_GETEDITSEL, CB_GETEXTENDEDUI, CB_GETITEMHEIGHT,
CB_GETLBTEXT, CB_GETLBTEXTLEN, CB_LIMITTEXT,
CB_RESETCONTENT, CB_SELECTSTRING, CB_SETCURSEL,
CB_SETEDITSEL, CB_SETEXTENDEDUI, CB_SHOWDROPDOWN:
As expected.

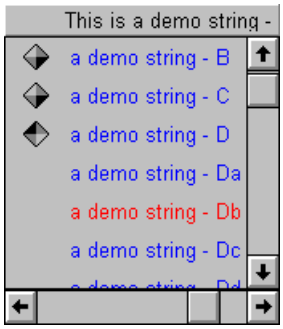
Notification Messages:

CZN_SELCHANGE:
Sent back to the parent window (usually the dialog box) whenever the item's selection has changed. Note that selection a section header will not provoke this message.
iMessage = WM_COMMAND
wParam = control window handle
lParam = MAKELONG(control id, CZN_SELCHANGE)
return value = not used

CZN_FILLSECTION:
Sent back to the parent window (usually the dialog box) whenever a section is about to be expanded and the control needs you to supply the items that belong to the section. When receiving this message you should send CZM_FILLSECTION messages back. (See CZM_FILLSECTION).
iMessage = WM_COMMAND
wParam = section identifier as specified in the CZM_ADDSECTION message. Watch out: unlike in most notification messages, here wParam is not the control's window handle.
lParam = MAKELONG(control id, CZN_FILLSECTION)
return value = not used

Default Behaviour:
The default style is CZCBS_SELECTED_9 | CZCBS_UNSELECTED_0 | CZCBS_BACKGROUND_15 | CZCBS_MARKER_1 | CZCBS_SIMPLE | CZCBS_DEMO.
The default caption (that is the demo text) is "Demo".

6. List box



Class Name:
The class name is "CZList"

Sizing:
The control implements the following size-related features:
1. The listbox has vertical an horizontal scroll-bars. Those scroll bars will appear or disappear dynamically whenever they are needed or not.
2. A margin is reserved on the left for the marker symbols. Markers are used to indicate the presence of section headers.
3. The listbox is handles tabs. By default, tab characters will expand to 8 times the average character with of the current font. Tab positions can still be set as desired by the `LB_SETTABSTOPS` message.
4. A caption is present on top of the list. The caption is single-line. Depending on the style you choose, the caption will or will not scroll horizontally with the list-box.
5. The listbox completely handles horizontal scrolling, still allowing the use of tabs. The margin reserved for the markers will be kept fixed.
6. The caption will be (dynamically) removed whenever the caption text is empty.

Font:
The control uses the same font as the parent window except it is not bold. In order to get this effect, you must instruct the parent window (usually the dialog box) to use a font that supports this selection, which the default font (Fixedsys) does not. Borland suggests you design your dialog boxes with the Arial 8 font.

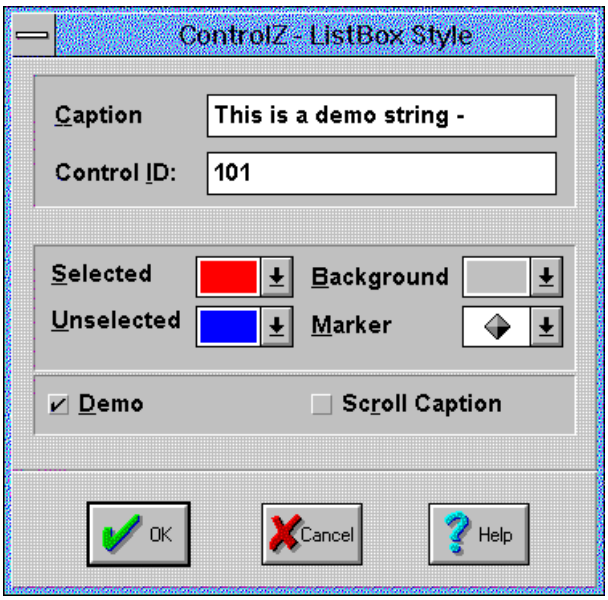
In the current version, there is no way to change the font. The `WM_SETFONT` message will have no effect. Still `WM_GETFONT` works fine.

The colors (background, foreground selected and unselected) for the listbox can be configured. See the styles section.

The caption background and color are not modifiable. The caption text will

appear in black on a raised gray background.

Styles:



The listbox control is much like the combo-box one. It supports the following styles, which can also be selected from the style dialog box shown above:

`CZLBS_SELECTED_0` to `CZLBS_SELECTED_15`: specifies the color for the item strings when selected.

In ControlZ, 16 predefined colors are used, coded from 0 to 15. To minimize differences that might occur when moving from one hardware configuration to the other, we don't rely on the system colors but define a proprietary 16-colors palette out of the most commonly used colors. Still it is most likely to just correspond with your system colors. Colors are coded as follows:

0 = black	1 = dark blue	2 = dark green	3 = dark red
4 = dark yellow	5 = dark pink	6 = medium blue	7 = light gray
8 = dark gray	9 = blue	10= green	11 = red
12 = yellow	13 = pink	14 = light blue	15 = white.

`CZLBS_UNSELECTED_0` to `CZLBS_UNSELECTED_15`: specifies the color for the item strings when not selected.

`CZLBS_BACKGROUND_0` to `CZLBS_BACKGROUND_15`: specifies the color for the list-box background.

`CZLBS_MARKER_0` to `CZLBS_MARK_3`: specifies the style of the marker. That is the symbol placed in the left margin. It is used to indicate that the item is either a section header or a normal item. When you select a section header, the previously expanded section will collapse and the selected section will

automatically expand to show its dependant items. The marker symbol is animated to indicate whether a section is currently expanded or not.

CZLBS_SCROLLCAPTION or CZLBS_FIXEDCAPTION: specifies whether the caption (if any) should be scrolled horizontally as the list box is scrolled.
As a general design consideration, you are advised to let the caption be scrollable in case you use tabs in the list-box and so have a multiple column concept. In that case you will indeed probable wish the column title to be horizontally aligned with the column itself. If you use no tabs (have only one column), then you may prefer to keep the caption fixed.

CZLBS_DEMO or CZLBS_NODEMO: specifies whether the control should act as a demo one. This feature has been added to allow you to design dialog boxes, run them and already have controls that "do something". That means that, while being in a preliminary study phase and having programmed nothing yet, you can show your boss or your customer more that a standard empty list-box.

The demo acting consists of generating 26 sections, each expanding to 26 items. This is done by appending letters ('A' to 'Z', then 'a' to 'z') to the control's caption.

By the time you want to use the control in your real application, you will have to remove the CZLBS_DEMO style. An alternative would be to delete all the demo sections after the control's creation (use the CZM_DELETESECTION message).

Messages:

WM_SETFONT:
Will have no effect.

WM_GETFONT and WM_SETTEXT:
As expected.

CZM_ADDSECTION:
Used at any time to add a section header to the combo-box. The section header will be appended at the end of the list.
wParam = a unique identifier for the section. You are responsible for providing this and checking uniqueness. Non uniqueness is not expected to be fatal but is not "documented". Allowed numbers range from 0 to 65535.
(LPSTR)lParam = section header string.
return value = index of the added string in the list or CB_ERR upon error.

CZM_DELETESECTION:
Used at any time to delete a section, regardless of whether it is currently expanded or not.

wParam = section identifier as specified in the CZM_ADDSECTION message.
lParam = NULL.
return value = number of remaining strings in the combo-box, or CB_ERR upon error. Note that the section identifier not being found is not considered as an error.

CZM_ISSECTION:
Used at any time to determine whether a given string in the list is a section header or a normal item.
wParam = index of the string in the list.
lParam = NULL.
return value = CB_ERR if an error occurs. Non zero if the wParam'th list item is a section header. Zero otherwise.

CZM_FILLSECTION:
Used when a CZN_FILLSECTION notification message is received. This is when a section is about to be expanded and the control needs you to supply the items associated with it. For that purpose you must send CZM_FILLSECTION messages back to the control, for each item that belongs to the section. Note that, when handling the CZN_FILLSECTION notification message, you should not send messages other than CZM_FILLSECTION. The section is considered to be filled when you return from the CZN_FILLSECTION notification message. Any attempt to send CZM_FILLSECTION messages outside the handling of CZN_FILLSECTION will have not effect.
wParam = NULL.
(LPSTR)lParam = item's string.
return value = index of the added string in the list, or CB_ERR upon error.

LB_ADDSTRING and LB_INSERTSTRING:
Provided for compatibility with the standard list-box. You actually can disable the section mechanism just by adding no section and using those messages instead. This will make the control to act as a standard list-box but will still provide you with the formatting capabilities (colors, caption, horizontal scrolling, etc.).
Using these two messages when you have sections in the list-box, may produce unpredictable results.

LB_RESETCONTENT, LB_SETTABSTOPS, LB_FINDSTRING, LB_GETCOUNT, LB_FINDSTRING, LB_GETCOUNT, LB_GETITEMRECT, LB_GETITEMHEIGHT, LB_GETCURSEL, LB_GETTEXT, LB_GETTEXTLEN, LB_DIR, LB_SELECTSTRING, LB_SETCURSEL, LB_GETSEL, LB_GETTOPINDEX, LB_SETTOPINDEX:
As expected.

LB_GETCARETINDEX, LB_SETCARETINDEX, LB_GETSELCOUNT, LB_GETSELITEMS, LB_SELITEMRANGE:

Will be ignored because the list-box does not have multiple selection capability.

Notification Messages:

CZN_SELCHANGE:

Sent back to the parent window (usually the dialog box) whenever the item's selection has changed. Note that selection a section header will not provoke this message.

iMessage = WM_COMMAND

wParam = control window handle

lParam = MAKELONG(control id, CZN_SELCHANGE)

return value = not used

CZN_FILLSECTION:

Sent back to the parent window (usually the dialog box) whenever a section is about to be expanded and the control needs you to supply the items that belong to the section. When receiving this message you should send CZM_FILLSECTION messages back. (See CZM_FILLSECTION).

iMessage = WM_COMMAND

wParam = section identifier as specified in the CZM_ADDSECTION message. Watch out: unlike in most notification messages, here wParam is not the control's window handle.

lParam = MAKELONG(control id, CZN_FILLSECTION)

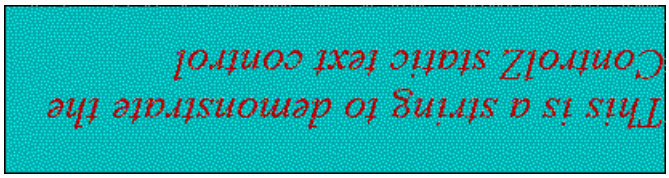
return value = not used

Default Behaviour:

The default style is CZLBS_SELECTED_9 | CZLBS_UNSELECTED_0 | CZLBS_BACKGROUND_15 | CZLBS_DEMO | CZLBS_MARKER_1 | CZLBS_SCROLLCAPTION.

The default caption (also used for the demo text) is "Demo".

7. Static Text



Class Name:
The class name is "CZText"

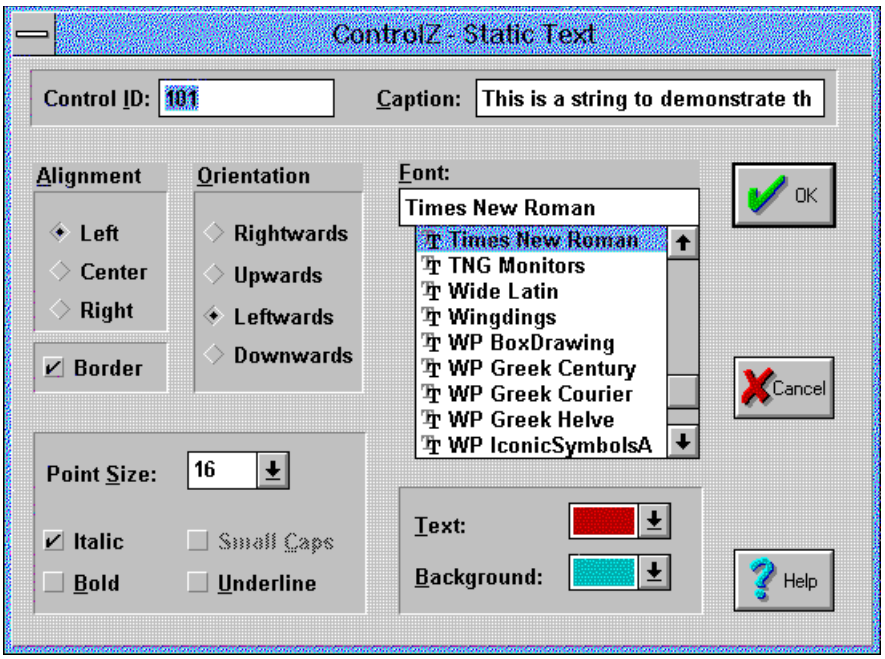
Sizing:
The static text control will basically resizes according to the styles specification (see the styles section). In addition the following general observations can be made:

1. The text will be split into several lines if necessary.
2. The text will be centred in the direction perpendicular to the writing direction. For example, normal text (writing left to right) will be centred vertically.
3. If the text cannot fit in the control's boundaries, some text will be clipped out, but the string will remain centred as described before.

Note that, as this is a static control, you will probably take care of sizing it correctly at the first time, when designing the resource.

Font:
The font handling is quite extensive and one of the main purposes of this control. See further the styles section.

Styles:



The static text control supports the following styles, which can also be selected from the style dialog box shown above:

`CZSTS_TEXTCOLOR_0` to `CZSTS_TEXTCOLOR_15`: specifies the text color.

In ControlZ, 16 predefined colors are used, coded from 0 to 15. To minimize differences that might occur when moving from one hardware configuration to the other, we don't rely on the system colors but define a proprietary 16-colors palette out of the most commonly used colors. Still it is most likely to just correspond with your system colors. Colors are coded as follows:

0 = black	1 = dark blue	2 = dark green	3 = dark red
4 = dark yellow	5 = dark pink	6 = medium blue	7 = light gray
8 = dark gray	9 = blue	10= green	11 = red
12 = yellow	13 = pink	14 = light blue	15 = white.

`CZSTS_BACKCOLOR_0` to `CZSTS_BACKCOLOR_15`: specifies the background color.

`CZSTS_LEFT`, `CZSTS_CENTER` or `CZSTS_RIGHT`: specifies whether the text should be left-aligned, centred or right-aligned, with respect to the writing direction. For instance, if the writing direction is upwards, then this will actually mean bottom-aligned, centred or top-aligned.

`CZSTS_RIGHTWARDS`, `CZSTS_UPWARDS`, `CZSTS_LEFTWARDS` or `CZSTS_DOWNWARDS`:
Specifies the writing direction. Rightwards is the normal direction like for example in this document. Leftwards is upside-down, etc. Note that the text will always remain centred in the perpendicular direction, no matter even the

size of the control.

CZSTS_UNDERLINE: specifies the (whole) text will be underlined.

CZSTS_BOLD: specifies the (whole) text will appear in bold characters.

CZSTS_ITALIC: specifies the (whole) text will be appear in italic characters.

CZSTS_SMALLCAPS: specifies the (whole) text will appear in upper case letters. The first letter of each word will be bigger than the others. In this case, the font size refers to the first letter of each word.
Please note that this style is provided for compatibility with future versions but is not actually implemented in this version of ControlZ.

Font typeface and size:
There are two ways to specify the font typeface and size:
1. Select in the style dialog box as shown above.
2. Don't use the style dialog box and specify it in the control's caption. In reality the caption text consists of:

- The font size in points (note: this is the usual unit, used in every word-processor).
- a '@' character.
- the typeface name.
- another '@' character.
- the text itself.

That is in summary (brackets indicate optional items):
`[size@][typeface@]text`
If the size and/or the typeface is missing, default values will be substituted, the default values being Fixedsys, 11.

Messages:
WM_SETFONT:
Will have no effect.

WM_GETFONT:
Will return the default font (usually the same as the dialog box), which is not the one actually used by the control.

WM_SETTEXT:
As expected. Note that the font size and typeface is expected to make part of the window text. So be careful with this message. See the styles section.

WM_GETTEXT:
As expected. Note that the font size and typeface usually makes part of the window text. So be careful with this message. See the styles section.

Notification Messages:
None.

Default Behaviour:
The default style is `CZSTS_CENTER | CZSTS_BACKCOLOR_15 | CZSTS_TEXTCOLOR_0 | CZSTS_RIGHTWARDS`.
The default font is Fixedsys, 11pt. Hence no underline, not bold, etc.
The default caption is `"Text"`.

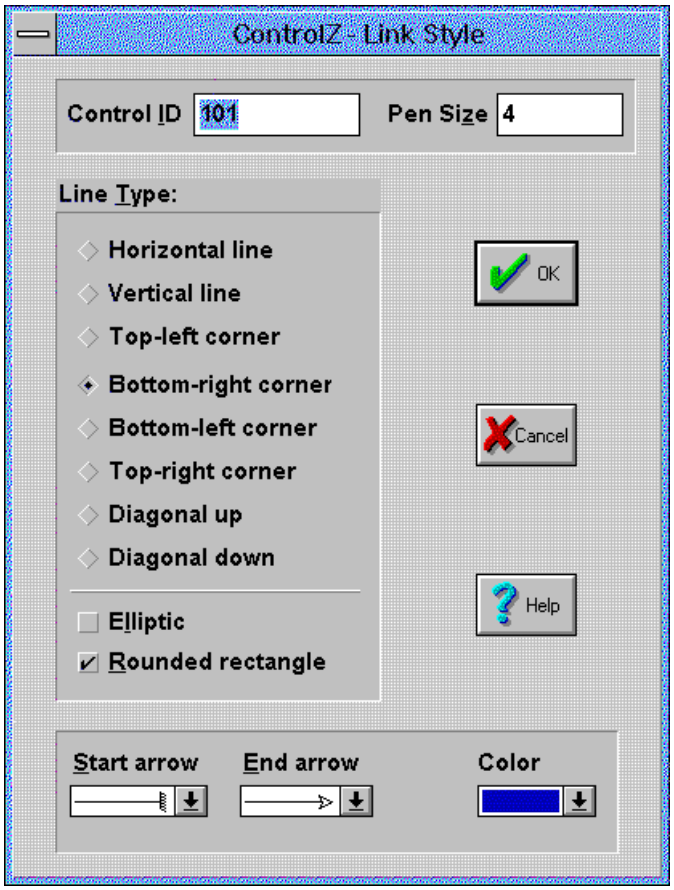
8. Static Link

Class Name:
The class name is "CZLink"

Sizing:
The link will always resize to be as big as the control size allows. That means both arrows are tangent to the control's edges, whatever the arrow style is.

Font:
The font is not relevant as the control contains no text.

Styles:



The static link control supports the following styles, which can also be selected from the style dialog box shown above:

CZLS_COLOR_0 to CZLS_COLOR_15: specifies the link color.

In ControlZ, 16 predefined colors are used, coded from 0 to 15. To minimize differences that might occur when moving from one hardware configuration to the other, we don't rely on the system colors but define a proprietary 16-colors palette out of the most commonly used colors. Still it is most likely to just correspond with your system colors. Colors are coded as follows:

0 = black	1 = dark blue	2 = dark green	3 = dark red
4 = dark yellow	5 = dark pink	6 = medium blue	7 = light gray
8 = dark gray	9 = blue	10= green	11 = red
12 = yellow	13 = pink	14 = light blue	15 = white.

CZLS_START_0 to CZLS_START_15: specifies the style for the start arrow. The is a choice of 16 arrow shapes, covering the most commonly used arrows. Special attention has been taken to provide arrows for structure charts (e.g. 1 to N relations, etc.)

CZLS_END_0 to CZLS_END_15: specifies the style for the end arrow.

CZLS_TLCORNER, CZLS_BRCORNER, CZLS_TRCORNER or CZLS_BLCORNER:
Specifies that the link should be curved to either the top-left, bottom-right, top-right or bottom-left corner.

CZLS_RECT, CZLS_ELLIPTIC, CZLS_ROUNDRECT or CZLS_DIRECT:
Specifies the kind of curving that is to be used. Namely a quarter of either a rectangle, an ellipse or a rounded rectangle.
The CZLS_DIRECT style implies that there is no curving. In that case, depending on the previous style choice, the result is as follows:
CZLS_TLCORNER | CZLS_DIRECT: gives an horizontal line, aligned at the top of the control.
CZLS_TRCORNER | CZLS_DIRECT: gives a vertical line, aligned at the left of the control.
CZLS_BLCORNER | CZLS_DIRECT: gives a diagonal line from the bottom-left to the top-right corner.
CZLS_BRCORNER | CZLS_DIRECT: gives a diagonal line from the top-left to the bottom-right corner.
To summarize and make those special cases easier to handle, the following styles have been defined:

```
#define CZLS_HLINE    (CZLS_TLCORNER | CZLS_DIRECT)
#define CZLS_VLINE    (CZLS_TRCORNER | CZLS_DIRECT)
#define CZLS_DIAGUP   (CZLS_BLCORNER | CZLS_DIRECT)
#define CZLS_DIAGDOWN (CZLS_BRCORNER | CZLS_DIRECT)
```

Pen size:

The pen size is specified in pixel units. There are two ways to do that:
1. Use the style dialog box as shown above.
2. Don't use this dialog box and specify the pen size is specified as the window text. In reality this is also the case when using the style dialog box.

There are a few things that you should keep in mind when specifying the pen size:
- a size of 0 is not allowed an will be replace by 1 automatically.
- the pen size should be an integer (do not use scientific notation, etc.).
- the pen size should be positive, although only the absolute value will be taken into account
- the pen size will be considered modulo 100. That means 101 is equivalent to 1.
- The arrows will be scaled proportionally to the pen size. Hence having a pen size bigger than a few pixels will result in sizing problems.

Messages:

`WM_SETFONT:`
Will have no effect.

`WM_GETFONT:`
Will return the default font (usually the same as the dialog box).

`WM_SETTEXT:`
As expected. Note that the window text is used to specify the pen size. So be carefull with this message. See the styles section.

`WM_GETTEXT:`
As expected. Note that the window text is used to specify the pen size. So be carefull with this message. See the styles section.

Notification Messages:

None.

Default Behaviour:

The default style is `CZLS_TLCORNER | CZLS_ELLIPTIC | CZLS_COLOR_0 | CZLS_START_0 | CZLS_END_1`.
The default caption (hence the pen size) is `"1"`.

9. How to register

The registration is 50 NLG (Dutch Guilders, < $30). See the `register.txt` file for more details. You will get a registration name and a password as soon as I have notification of payment from the bank. Note that there may be some delay between the time you pay and the time I receive notification of that.
This is a per-copy price. Thus if you will have copies of ControlZ on N stations in your orgranisation, then please register N copies.

Commercial use of ControlZ
If you intend to make commercial use of ControlZ, you may want either to:
- register as many copies as needed, or
- get the source code, in which case you should contact me first.