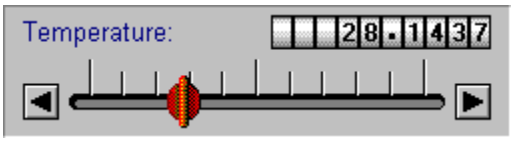


ControlZ 1.31 Custom Control Library Programmer's Regerence.

Author's E-Mail: cvzwynsv@estec.esa.nl

Table of contents

Control	Class	Sizing	Font	Style	Msg.	Notif.	Default
Scaler	"CZScaler"						
Dial	"CZDial"						
Tuner	"CZTuner"						
List Box	"CZList"						





Combo Box "CZCombo" 





Static Text "CZText" 




none


Arrowed Link "CZLink"  none



none


What's new ?

- The frist released version is ControlZ 1.2
- Version 1.3 adds the run-time libraries.
- Version 1.31 fixes all known bugs, has a better color management and display speed for the Tuner control. Also the help file has been reviewed.

ControlZ - Introduction

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 Files | 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.

ControlZ implements 7 types of controls::

- 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.

The ControlZ.DLL that comes in the package 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 help file (this file!).

As from version 1.3 there is also a collection of run-time libraries. The run-time libraries do not provide resource editing features, but do give you the right to distribute them along with your application. Also, since there is one run-time library for each type of control, you pay only for what you need.

See also:

Programming principle.

Programming Principle

Note: In this help file, I assume you are programming in **C**. Though I have not tried to do so, it should be possible to use ControlZ also with **C++**, **Pascal**, or whichever language that is able to link to a DLL. This is because you communicate with the controls only via Windows messages, so the only thing you need is the *SendMesssage* function, which is the minimum any language should provide !

All right, for the run-time libraries (see below), you have to call the `CZRegister...` functions so there I guess you will have to slightly modify the header file.

1- Instal ControlZ

A good idea to begin with is to copy the DLL's and the help file to your windows (or windows\system) directory.

In either of the following cases:

- you have renamed the ControlZ.hlp file, or
- the controlZ.hlp file is located neither in the windows (or windows\system) nor in the current directory;

you will have to include the following lines in your WIN.INI file:

```
[ControlZ]
HelpFile=<full name and path of this help file>
```

Now, in the Borland Resource Workshop, choose `Files | Install Control Library` and install ControlZ.DLL. You will then get seven additional buttons in the tools window when you are editing a dialog box. Use them to add ControlZ controls to your dialog boxes. Double clicking on one control will bring up a style dialog box so you can configure the control.

If you don't have Resource Workshop, you will probably have to edit your resource file (`*.RC`) by hand.

2 - Create ControlZ controls

The easiest way to include ControlZ controls in your application is to edit your dialog boxes with the Borland Resource Workshop as suggested above.
You may also create controls in your application by supplying the correct **class name** and window **style** to window creation functions (eg. *CreateWindow*) Those are completely described in this help file.

In order for the controls to be available to your application, you have to load the ControlZ library at the beginning of your program. Here is an example of how to do that:

```
int PASCAL WinMain (HINSTANCE hInst, HINSTANCE hPrev,
                    LPSTR lpszCmdLine, int nCmdShow)
    {
    WORD hLibCZ;

    hLibCZ = LoadLibrary("CONTROLZ.DLL");
    if (hLibCZ < 32)
        {
        MessageBox(NULL, "Can't load ControlZ.DLL.",
                   szAppName, MB_OK);
        return -1;
        }
```

```
/*
 * Your code here ...
 */
FreeLibrary(hLibCZ);
}
```

3- Control the controls

Assuming you have created the controls with the desired styles and don't want to dynamically change those styles, your program will exclusively communicate with the controls via Windows **messages**. That is, you call the *SendMessage* function (or equivanlent) to send messages to the controls. Available messages are completely described in this help file.
In some cases, the controls notify their parent window (usually a dialog box) about an action that has taken place. The parent windows receive those **notification messages** in the high-order word of the `lParam` parameter of the `WM_COMMAND` message. Notification messages are fully described in this help file.

4- Working with the run-time libraries

The run-time libraries implement the same functionnality as the original ControlZ.DLL, except that there is no resource editing feature (ie. no link with Borland's Resource Workshop).
There is one run-time library for each type of control. Run-time libraries are registered separately from the original ControlZ library, but they have the advantage that **you may distribute them with your application**.

When your application is finished, you can link it to the ControlZ run-time libraries by proceeding as follows:

- When you register the run-time libraries, you will get a name and password for each one. This is not the same as for the ControlZ.dll library.
- Include the "`runtime.h`" file to your source files.
- Add the following (relevant) lines to your project's .DEF file:

```
IMPORTS
      RegisterCZScaler=CZScaler.1
      RegisterCZDial=CZDial.1
      RegisterCZTuner=CZTuner.1
      RegisterCZList=CZList.1
      RegisterCZCombo=CZCombo.1
      RegisterCZText=CZText.1
      RegisterCZLink=CZLink.1
```

  This will automatically load the DLL's so you don't have make any *LoadLibrary* call.

  Note that there are alternatives to this. One of them is to make and import library. Your compiler should provide a tool for that (IMPLIB, etc.). Still, I personally find the `IMPORT` statement rather simple and portable.

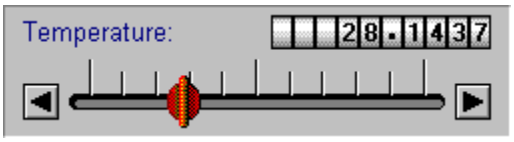
- In your program's *WinMain* function, remove the *LoadLibrary("controlz.dll")* and *FreeLibrary* calls described above. Instead make as many calls to the `RegisterCZ...` functions as relevant. Those function have all the same prototype, as declared in the "`runtime.h`" header file.:

```
BOOL FAR PASCAL RegisterCZ<control>(LPSTR szName, LPSTR szCode)
```
szName = Your registration name.
szCode = Your registration password.
return vvalue = Non-zero if it's okay.

It is recommended that you do not put those two strings in clear in your source code...

- Make a *case insensitive* link and import.

ControlZ - Scaler



The scaler control is used to let the user choose a number (long integer or floating point), within a given range. It can also be used without user interaction, for eample to display percentage data.

Class Name : `"CZScaler"`
Sizing
Font
Styles
Messages
Notification messages
Default Behaviour

Scaler - 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.

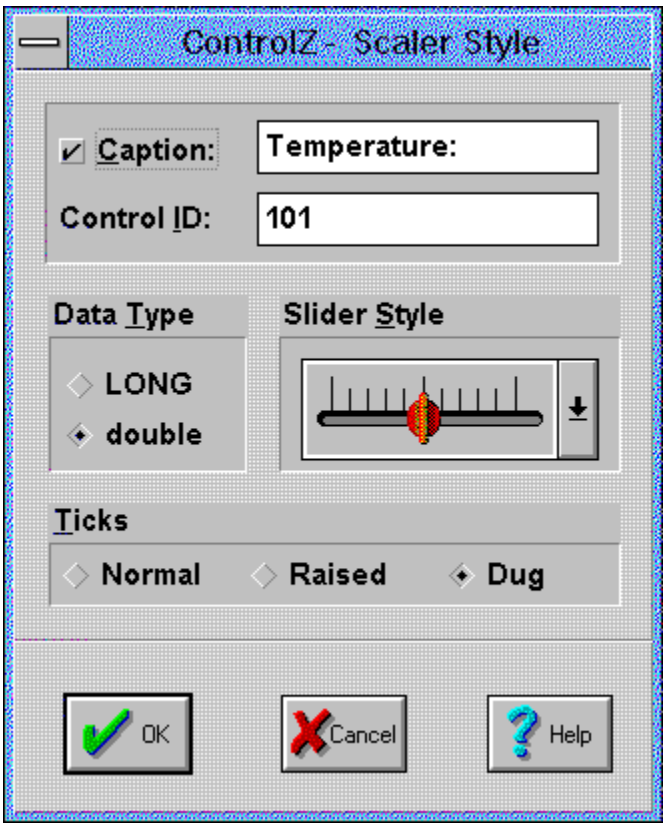
Scaler - Font

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.

Scaler - 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.

Scaler - 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
  lParam: (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
  lParam: (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.
  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
  lParam: (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
  lParam: (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.
  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
  lParam: (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
  lParam: (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.
  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

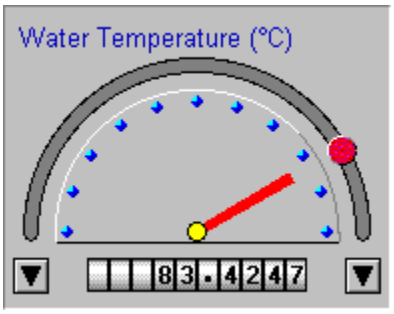
Scaler - Notification Messages

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

Scaler - 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`.

ControlZ - Dial



The dial control is used to let the user choose a number (long integer or floating point), within a given range. It can also be used without user interaction, for eample to display percentage data. The user interacts with the control by either dragging the slider, or pressing the left or right buttons. The slider color is animated when the value changes.

Class Name: `"CZDial"`

Dial - 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.

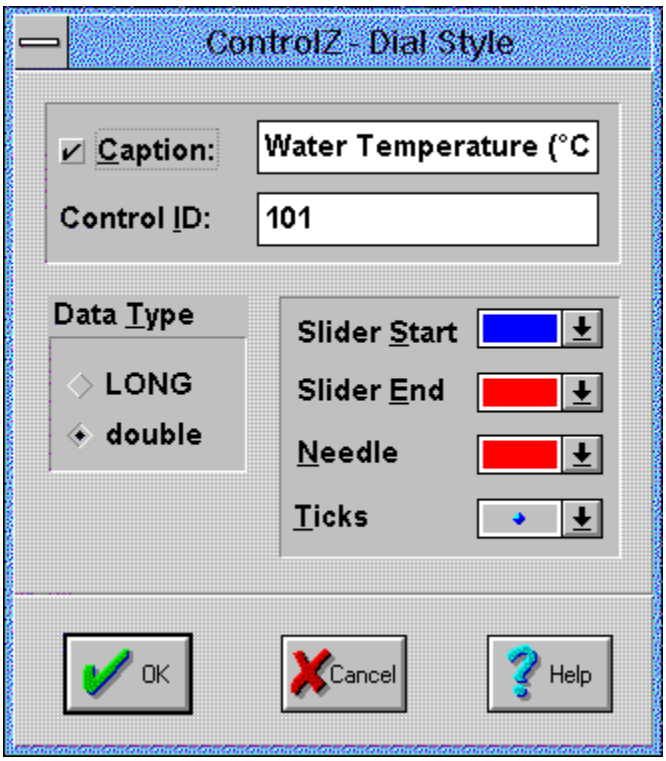
Dial - Font

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.

Dial - 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 posi tion.
- `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.

- `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.

Dial - 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
  lParam: (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
  lParam: (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.`
  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
  lParam: (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
  lParam: (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.`
  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
  lParam: (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
  lParam: (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.
  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

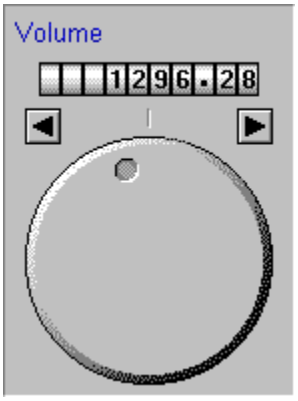
Dial - Notification Messages

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

Dial - 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`.

ControlZ - Tuner



The tuner control is used to let the user choose a number (long integer or floating point), within an **unlimited range**. The user interacts with the control by either turning the wheel (ie. dragging the slider), or pressing the left or right buttons. Your application can, in particular, specify the range given to one wheel turn.

Class Name: `"CZTuner"`
Sizing
Font
Styles
Messages
Notification messages
Default Behaviour

Tuner - 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.

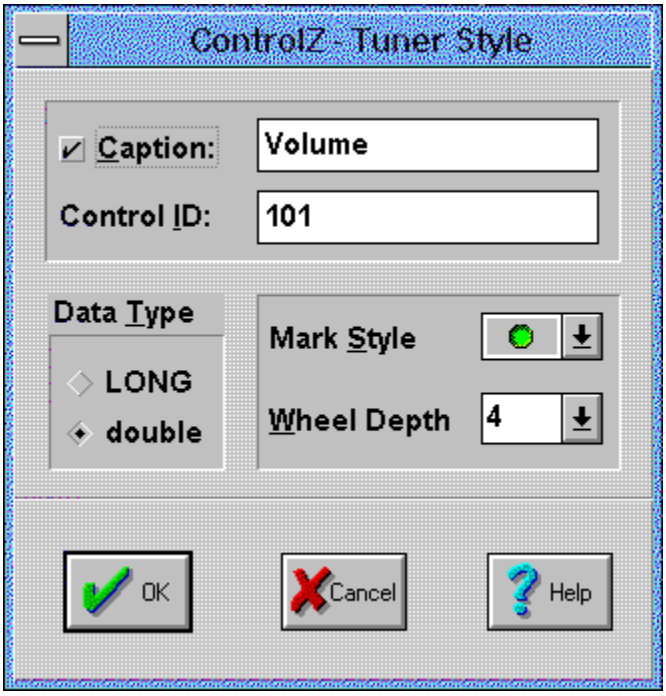
Tuner - Font

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.

Tuner - 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.

Tuner - 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
  lParam: (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
  lParam: (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.
  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
  lParam: (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
  lParam: (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.
  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
  lParam: (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
  lParam: (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.
  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

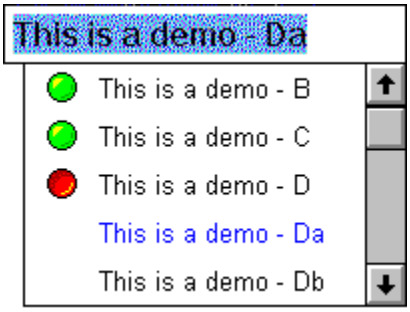
Tuner - Notification Messages

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

Tuner - Default Behaviour

- The default style is `CZTS_CAPTION | CZTS_TYPELONG | CZTS_WHEELDEPTH_4 | CZTS_MARK_1`.
- The default caption is `"Volume"`.
- The default range is 0 to 100.
- The default increment is 1.

ControlZ - Combo Box



The combo-box control is used to let the user choose an item out of a list when you don't want or don't have the place to display the list permanently. In addition, the ControlZ combo-box has a **one-level hierarchy** (top level entries are called sections), **where only one section can be open at a time**. This makes it possible to display virtually very long lists without overloading both the user and the memory. Note that standard windows list- and combo-boxes are limited to a few thousand items.

A common solution to this limitation is to use a virtual list-box. This has the disadvantage that the user has to scroll very much before he gets the desired item. Things get even more tedious when the number of items is not known in advance and one cannot scroll correctly.

Another approach is a hierarchical list-box. The disadvantage of this method is that the number of items is limited again to a few thousand, and that the hierarchy has to be well defined and known by the user.

In summary, the ControlZ combo-box **takes advantage of both the virtual and hierarchival list-box**. Also note that those two techniques are widely spread for list-boxes but I don't know of any **implementation for combo-boxes**.

> One very good example (I guess) of the use of the ControlZ combo-box is shown in the GraphZ.DLL library (Copyright © 1995, C. van Zwynsvoorde.) available from the CICA anonymous FTP server.
> GraphZ uses such a combo-box to implement a date & time selection mechanism. The hierarchy and number of items cannot be known in advance. The number of items is virtually more than 60 billion. Still the selecttion of a date & time item is done in a few mouse clicks.

In addition, the ControlZ combo-box has:

- a "**demo**" attribute which makes it be "alive" without having to do anything and even when your are editing the resources. Indeed, what I don't like about standard window combo-boxes is that they are always empty by default !
- extended **color attributes** included in the styles, so you don't have to care at all about that and this is also visible at resource edition time.
- compatibility with the standard combo-box if you don't want to use sections.

Class Name: `"CZCombo"`
Sizing
Font
Styles
Messages
Notification messages
Default Behaviour

Combo Box - 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.

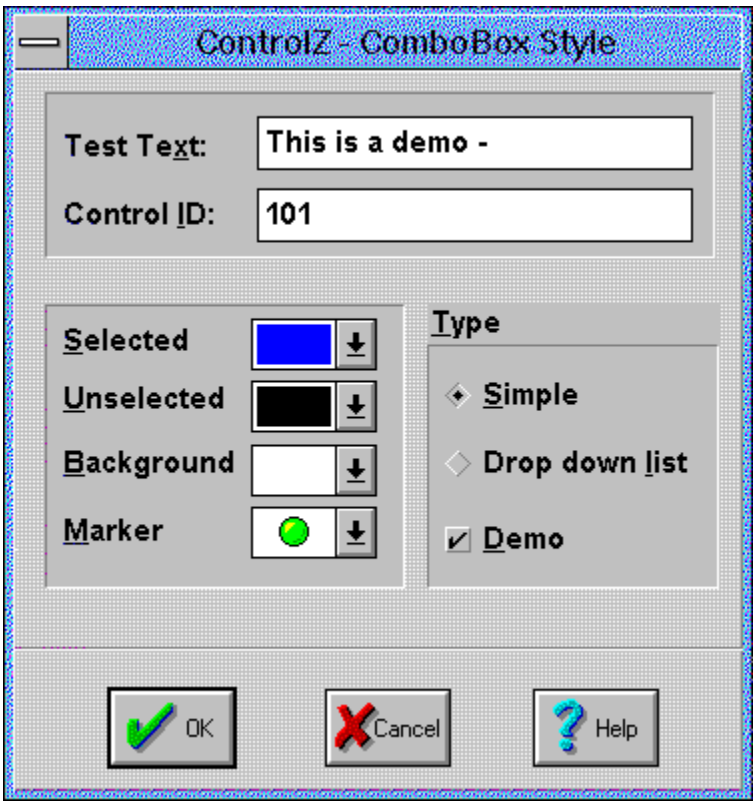
Combo Box - Font

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.

Combo Box - 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.

`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).

Combo Box - 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.
  lParam: (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.
  lParam: (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_GETCURSEL,CB_GETDROPPEDCONTROLRECT, CB_GETDROPPEDSTATE, CB_FINDSTRING, CB_GETCOUNT,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.

Combo Box - Notification Messages

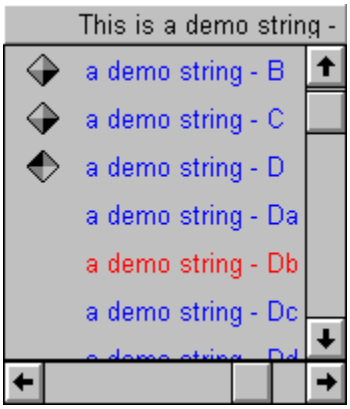
- CZN_SELCHANGE:
  Sent back to the parent window (usually the dialog box) whenever the item's selection has changed. Note that selection a section header will not provoke this message.
  iMessage: WM_COMMAND
  wParam: control id.
  lParam: MAKELONG(control window handle, CZN_SELCHANGE)
  return value: 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: contro lid.
  lParam: MAKELONG(section, CZN_FILLSECTION). The low-order word is the section identifier as specified in the CZM_ADDSECTION message. Watch out: unlike in most notification messages, here LOWORD(lParam) is not the control's window handle.
  return value: not used

Combo Box - 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`".

ControlZ - List Box



The ControlZ list-box control has all the features of the ControlZ combo-box. In addition it has a **title**, and fully implements **horizontal scrolling**. It also supports **tabbing**, which is interesting when you want to display arrayed data. Finally, it has the ability to horizontally **scroll the title** along with the list-box so that you can get a column effect.

Note: Though it was not primarily designed for that purpose, you can use the tabbing and color features (let the selected and unselected colors be the same) to simulate **a display-only grid** control.

Class Name: `"CZList"`
Sizing
Font
Styles
Messages
Notification Messages
Default Behaviour

List Box - 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.

List Box - Font

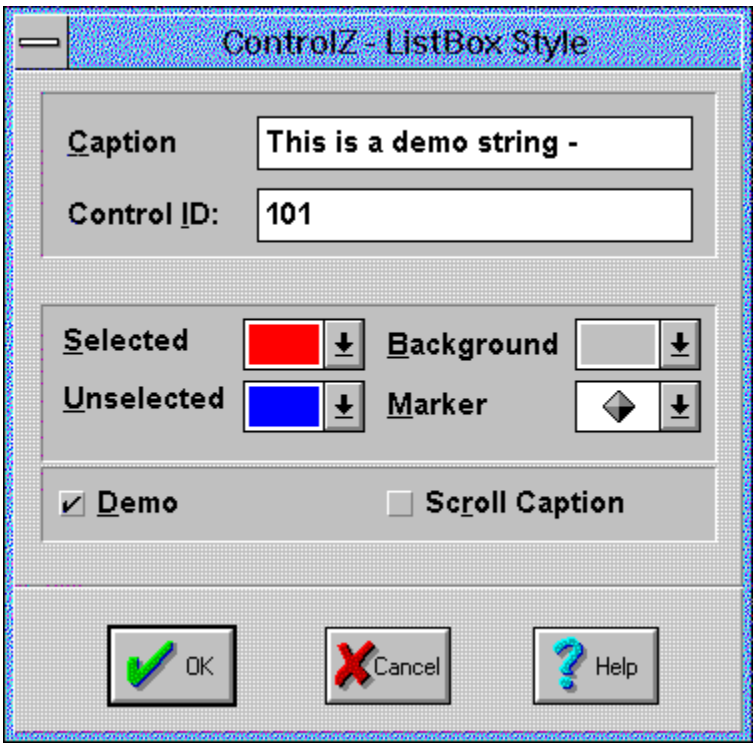
The control uses the same font as the parent window except it is not bold. In order to get this effect, you must instruct the parent window (usually the dialog box) to use a font that supports this selection, which the default font (Fixedsys) does not. Borland suggests you design your dialog boxes with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

List Box - 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.
- `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).

List Box - 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.
  lParam: (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.
  lParam: (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.

List Box - Notification Messages

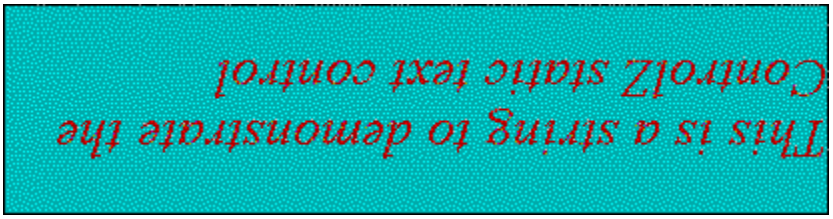
- CZN_SELCHANGE:
  Sent back to the parent window (usually the dialog box) whenever the item's selection has changed. Note that selection a section header will not provoke this message.
  iMessage: WM_COMMAND
  wParam: control id.
  lParam: MAKELONG(control window handle, CZN_SELCHANGE)
  return value: 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: control id.
  lParam: MAKELONG(section, CZN_FILLSECTION). The low-order word is the section identifier as specified in the CZM_ADDSECTION message. Watch out: unlike in most notification messages, here LOWORD(lParam) is not the control's window handle.
  return value: not used

List Box - 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`".

ControlZ - Static Text



The ControlZ static text control is meant to display static text, probably in a dialog box. because of the extended text formatting features, you can visually improve your dialog boxes and don't have to do this (rather tedious) work inn your program. Formatting attributes are: the **font name and size**, **bold**, **italic**, **underline**, the **vertical and horizontal justification**, the **color** and the **text direction**. The control also looks well with a border.

Class Name: `"CZText"`
Sizing
Font
Styles
Messages
Notification Messages: none.
Default Behaviour

Static Text - Sizing

The static text control will basically resize according to the style specification. In addition the following general observations can be made:

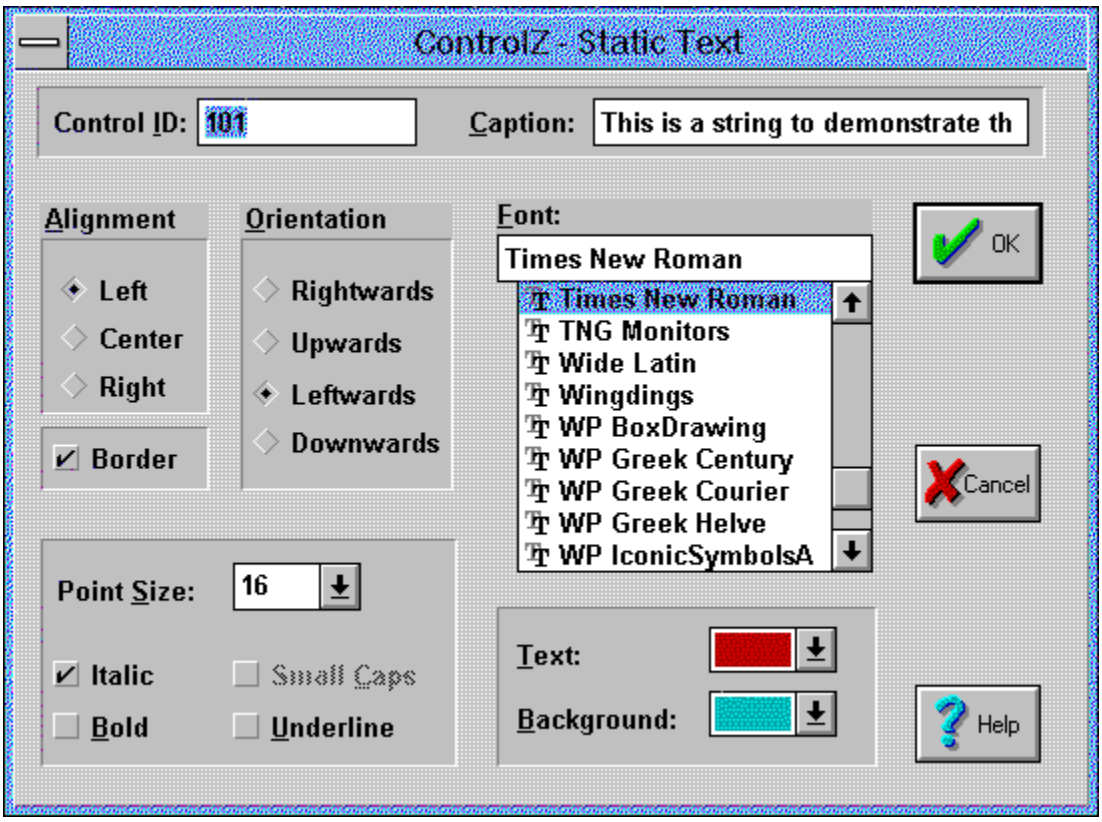
1. The text will be split into several lines if necessary.
2. The text will always be centred in the direction perpendicular to the writing direction. For example, normal text (writing left to right) will be centred vertically.
3. If the text cannot fit in the control's boundaries, some text will be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

Also 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.

Static Text - Font

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

Static Text - 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.
- `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.

Static Text - 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.

Static Text - 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`".

ControlZ - Static Link

The ControlZ static link control amkes it easy to display all kind of **static arrowed links** in your dialog boxes and so improve their visual quality. The control has a transparent background so it does not affect other controls. The control also has some **color** support. The most commonly used arrows have been implemented, along with some symbolic links and a few links I personnlly use for database entity relationship diagrams. Note that the control is static: there is no logical attachment to other objects.

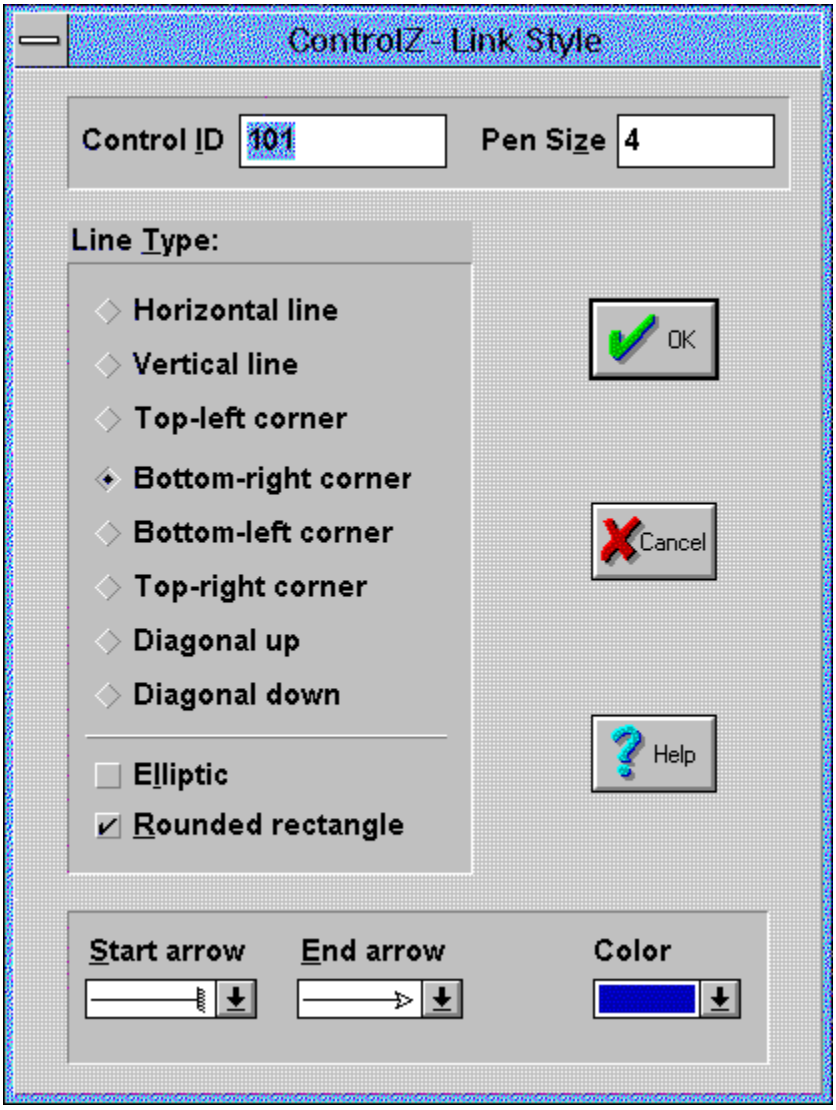
Note that drawing arrows with only the Windows API is a tedious thing to do !

Class Name: `"CZLink"`
Sizing
Font: not relevant as there is no text.
Styles
Messages
Notification Messages: none.
Default Behaviour

Static Link - 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.

Static Link - 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.
- `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.

Static Link - 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.

Static Link - 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`".

ControlZ - How to Register

There are two different kind of registrations (well, actually there are three):

1 **The ControlZ.dll library.**
The registration fee amounts US$ 45.
This will remove all the "`unregistered copy`" remainders.
You are not allowed to distribute this library, unless you register one copy per user.

2 **The run-time libraries (CZxxxx.dll).**
The registration fee amounts US$ 45 per library.
You have to register each of them separately. That is, you pay only for what you need.
You are allowed to distribute them along with your apprication as many time as you want, since the registration is made inside your program.

3 **The complete suite.**
This includes both registrations 1 and 2.
The fee is discounted to US$ 195.

When you register, you will get a name and password for each library. Non of them use the same.

You will get nothing more (argh!) since the distributed libraries are already fully featured and the full documentation consists of this help file.

In order to register, please proceed as indicated below:

- Fill in the registration form.

- Send it to:
```
C. van Zwynsvoorde.
Zeestraat 21,
2201 KH Noordwijk.
The Netherlands.
```
or fax it to:
```
(+31) 1719-85659
```

- Transfer the registration fee to the following account:
```
Bank account number: 56.79.21.395
Bank: ABN-AMRO bank in The Netherlands.
Bank account owner: C.S van Zwijnsvoorde.
```
I'm sorry I can't take credit cards.
Possibly e-mail me when you have done so, so that I can check it more quickly.

- As soon as I have notice of your payment from the bank, I will send you back your name(s) and password(s).

Questions:

If you have any question, remark, desire bug report or if you want to get the source code, please e-mail me at `cvzwynsv@estec.esa.nl`

See also:

Programming principle.

ControlZ 1.31 Registration Form

To register ControlZ 1.31, please first deposit the correct sum on the bank account number 56.79.21.395 of the ABN-AMRO bank in the Netherlands under the name "C.S van Zwijnsvoorde". Then fax or mail me a printed copy of this page. As soon as I have notification of your payment from the bank you will be retruned a registration name and a password, along with installation instructions.

Terms/Conditions:

I agree that ControlZ is distributed as shareware. No warranty exists, either express or implied. No liability is assumed for any damage or loss resulting from the use of this program. No claims are made regarding the accuracy of this program. The author reserves the right to change pricing in future versions.
I agree that ControlZ must be licenced for each user.
The ControlZ run-time libraries only have to be licenced by the developper. Your signature below indicates your acceptance of these terms and conditions.

Product	Price (US$)
O ControlZ.DLL	_____ (45)
O CZScaler.DLL	_____ (45)
O CZDial.DLL	_____ (45)
O CZTuner.DLL	_____ (45)
O CZList.DLL	_____ (45)
O CZCombo.DLL	_____ (45)
O CZText.DLL	_____ (45)
O CZLink.DLL	_____ (45)
O Complete suite	_____ (195)
TOTAL	_____

User Name: _____
(Name you would like to have as your User Name. For the run-time libraries this is recommended to be your application's name)

Your Complete Name:
_____

Postal Address:
_____

_____

City: _____ Country: _____

Phone No: _____ Fax No: _____

Internet E-mail address: _____

How do you prefer to receive your user name and password ?
O. E-Mail O. Post O. Fax

Signature: _____ Date: _____

****
* Mail address: C.v.Zwynsvoorde. Zeestraat 21. *
* 2201 KH Noordwijk zh. The Netherlands *
* Fax number: (+31) 171985659 *
****

Colors:

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.