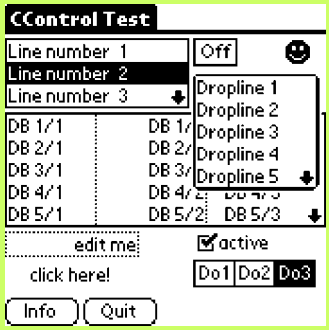
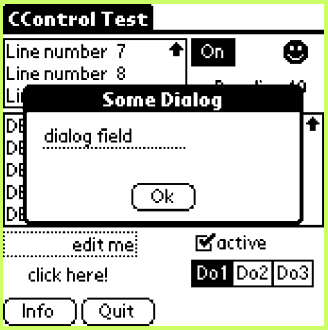
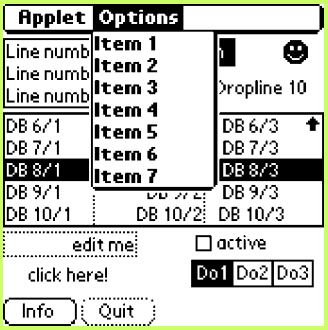
CControls and CEditor

Introduction

Ever thought it is impossible to create real palm pilot applications using PocketC?
CControls is a function library used to develop graphic front ends on your palm pilot with PocketC, offering all GUI controls commonly known on the pilot platform - and even more.
The library comes bundled with a WYSIWYG-editor with which you're able to design directly on the palm pilot.
Please read the "Features" sections to get an idea what CControls and CEditor offer you.







Installation

The package contains the following files:

- Ceditor.prc: CEditor application files
- CcontrolsDB.pdb: CControls function library database
- Ccontrols example.c: CControls code example
- Manual.html: this manual

Please install the PRC- and PDB-files on your palm pilot using the "install" function of the pilot desktop.
The function code for CControls (three source memos) is stored in the library database. To be able to use CControls in your own code, you first have to call CEdtor and extract this code by executing "Install library" under the "Form"-menu.
After installation you will find three new memos:
/$ Ccontrols.c
/$ Ccontrols1.c
/$ Ccontrols2.c

These three source code files will be included every time you generate code for forms using CEditor. CEditor will automatically check if the libray is missing and re-extract it when generating code.
If you‘re coding your own applications without using CEditor, please take care to include the library code (look at the CControls code example).
Important: CControls and CEditor only work when PocketC (version 3.04 or higher) is installed on you palm pilot. PocketC is not included in this package.

Features

CControl features

- supports all common pilot controls
- supports tables and database-linked tables
- supports real menus
- supports dialog frames
- all controls have palm pilot's standard outfit and behaviour

- customizable outfit for certain controls like edit fields
- all controls are created dynamically (the number of used controls is only limited by your pilot's memory)
- all control's contents are created dynamically (the number of added items is only limited by your pilot's memory)
- easy to use: no need to initialize special environments
- a small footprint (controls library uses about 12k / three memos)

CControls GUI controls

Currently, CControls supports the following controls:

- text buttons
- image buttons
- switch buttons
- edit fields
- checkboxes
- radio buttons
- text labels
- popups
- listboxes
- tables
- database linked tables
- menus
- dialog frames

CEditor features

- design GUI applications directly on your Palm Pilot
- WYSIWYG front end
- supports all controls (apart from database linked tables)
- supports saving form projects
- generates source code for complete PocketC applications (including framework, event handling and method functions to be filled with your own control actions)
- different modes while designing: move, size, edit and test your controls
- supports editing of controls not directly visible (popups and menus)
- supports filling controls with items (e.g. for listboxes)
- supports defining your own variable names for code generation
- easy to use form editor for designing exact GUI front ends
- was written in PocketC using CControls

CControl Usage

It is very simple to use CControls in your own PocketC source code:

- switch to graphic mode:
  graph_on()
- allocate a handler for every control:
  Chandle mylistbox;
- create a control by defining it‘s position and outfit:
  mylistbox = Clistbox(10,10,80,60);
- use the handle to modify the control:
  Csetrow(mylistbox, 0, 30, 1);
- use the handle to fill the control with items:
  Cadditem(mylistbox,"First Item");
- use the handle to get certain properties:
  size = Cgetcount(mylistbox);

- use the handle to set a certain position:
  Csetposition(mylisbox, 1);
- use the handle to access the control:
  current = Cgetcursel(mylistbox);
  result = Cgetcontent(mylistbox);
- use the handle to catch events thrown by the control:
  if Cevent(mylistbox) ...
- use the handle to destroy the control if no longer needed:
  Cdestroy(mylistbox);

Always remember:

- all controls are created dynamically
- all items are added dynamically
- there are no limitations (apart from the amount of memory)

If you want to learn more about the usage of CControls in your own code, please refer to the sample code. Another quick approach is to use CEditor to design your GUI front end, let it generate the PocketC code for you and fill the (empty) event functions with your own actions.

CControl code example

```
// CControls example.c
include "Ccontrols.c"

// control handles
Chandle hl1, he1, hlb1, ht1, hm1, hm2, hmf, hb1,
hb2, hi1, hd1, hp1, hs1, hc1, hr1;
pointer DB;


initcontrols(){
int i,r;

// button 1
hb1=Cbutton(1,147,35,0,1,4);
Csetcontent(hb1,"Info");

// button 2
hb2=Cbutton(40,147,35,0,1,4);
Csetcontent(hb2,"Quit");

// image-button 1
hi1=Cimage(140,18,12,12);
Csetcontent(hi1,
"0c1f83fc7fee67e67ffffffdfbef770e3fc1f8");

// listbox 1
hl1=Clistbox(1,18,90,3);
for(i=1;i<=10;i++) Cadditem(hl1, "Line number "+i);
Csetcursel(hl1,1);

// dropdown 1
hd1=Cdropdown(95,36,60,5);
for(i=1;i<=10;i++)
Cadditem(hd1, "Dropline "+i);

// popup 1
hp1=Cdropdown(14,23,70,5);
for(i=1;i<=8;i++)
Cadditem(hp1, "Popline "+i);

// switch 1
hs1=Cswitch(95,18,20,12,1,0);
Csetcontent(hs1, "Off");

// checkbox 1
hc1=Ccheckbox(95,113,40);
Csetcontent(hc1, "active");
Csetstate(hc1, 1);

// edit field 1
he1=Cedit(1,114,65,2,2,2);
Csettopic(he1, "Enter text:");
```

```
createDB(){
int i,r; pointer p;
p=malloc(3); settype(p,3,'s');
if (!dbcreate("CTestDB")){
alert("\nUnable to create CTestDB.");
return;}
for(i=0;i<10;i++){
for(r=0;r<3;r++){
p[r]="DB "+(i+1)+"/"+(r+1);
}
dbrec(-1);
dbwritex(p,"szszsz");}
}


initDB(){
DB = malloc(3);
settype(DB, 3, 's');

// create database
if(!dbopen("CTestDB"))
createDB();

// create database-table
ht1=CtableDB(1,55,158,5,3,DB,"szszsz");
Csetrow(ht1,0,45,0);
Csetrow(ht1,1,55,2);
Csetrow(ht1,2,48,1);
}


// actions on controlevents:
onbutton1(){
alert("Listbox:"+Cgetcontent(hl1)
+"\nSwitch:"+Cgetstate(hs1)+"\nRadio:"+Cgetcursel
(hr1));
}


onlabel1(){
dialog(); drawscreen();
}


onimage1(){
if (Cpopupevent(hp1))
alert("Popup:\n"+Cgetcontent(hp1));
}


onswitch1(){
if (Cgetstate(hs1))
Csetcontent(hs1, "On");
else
```

```
Csetcontent(he1, "edit me");

// label 1
hlb1=Clabel(1,129,65,0,2,1);
Csetcontent(hlb1, "click here!");

// radio 1
hr1=Cradio(95,128,60,0);
for(i=1;i<=3;i++)
Cadditem(hr1, "Do"+i);

// menu-topic 1
hm1=Cmenu(5,60,40);
Csettopic(hm1,"Applet");
for(i=1;i<=5;i++) Cadditem(hm1, "Item"+i);

// menu-topic 2
hm2=Cmenu(45,60,45);
Csettopic(hm2,"Options");
for(i=1;i<=7;i++) Cadditem(hm2, "Item "+i);

// menu-bar
hmf=Cmenubar();
Caddmenu(hmf, hm1);
Caddmenu(hmf, hm2);
}

drawscreen(){
clearg();
title("CControl Test");

// draw controls
Cdraw(hb1); Cdraw(hb2);
Cdraw(hl1); Cdraw(hi1);
Cdraw(hd1); Cdraw(hs1);
Cdraw(hc1); Cdraw(he1);
Cdraw(hlb1); Cdraw(hr1);
Cdraw(ht1);
}

dialog(){
Chandle hf1, he1, hb1; int e;

// dialog frame
hf1=Cframe(10,40,140,70);
Csetcontent(hf1, "Some Dialog");

// edit field 1
he1=Cedit(20,60,70, 2,1,0);
Csetcontent(he1, "dialog field");

// button 1
hb1=Cbutton(65, 90, 30, 12, 1, 4);
Csetcontent(hb1, "Ok");

// draw dialog
Cdraw(hf1); Cdraw(he1); Cdraw(hb1);

// message loop
while(1){
e=event(1);
if (Cevent(he1,e));
else if (Cevent(hb1,e))
break;}
}
```

```
Csetcontent(hs1, "Off");
Cdraw(hs1);
}

oncheckbox1(){
if (Cgetstate(hc1)) Cactivate(hb2);
else Cdeactivate(hb2);
}

onmenu(string s){
alert(s);
}

main(){
int e;

// initialize screen
graph_on();
title("Please wait....");

// open DB and init tableDB
initDB();

// initialize controls
initcontrols();

// display main-screen
drawscreen();

// message loop
while(1){
e=event(1);
if (Cevent(hb1,e))
onbutton1();
else if (Cevent(hb2,e))
break;
else if (Cevent(hl1,e));
else if (Cevent(hd1,e));
else if (Cevent(hi1,e))
onimage1();
else if (Cevent(hs1,e))
onswitch1();
else if (Cevent(hc1,e))
oncheckbox1();
else if (Cevent(he1,e));
else if (Cevent(hlb1,e))
onlabel1();
else if(Cevent(hmf,e))
onmenu(Cgetcontent(hmf));
else if(Cevent(hr1,e));
else if(Cevent(ht1,e));
}

// close DB
dbclose();
}
```

CControl functions reference

Creation functions

Cedit(x,y,w,l,s,a)		
x y w l	x-position y-postion width line: 0 = white 1 = black	Creates a new edit field at the given position with the given style. Edit fields prompt for input via dialog when clicked.

	2 = gray	
s	style: 0 = plain 1 = underlined 2 = boxed	
a	alignement: 0 = left 1 = center 2 = right	
Clabel(x,y,w,l,s,a)		
x y w l	x-position y-postion width line: 0 = white 1 = black 2 = gray	Creates a new label field at the given position with the given style. Label fields fire events when clicked, so that they can be linked to an action if necessary.
s	style: 0 = plain 1 = underlined 2 = boxed	
a	alignement: 0 = left 1 = center 2 = right	
Clistbox(x,y,w,h)		
x y w h	x-position y-postion width height	Creates a new listbox at the given position with the given style. Height is defined in items. Listboxes can be filled with items using *Cadditem*. To access certain items you can use *Csetcursel* and *Cgetcontent*.
Cdropdown(x,y,w,h)		
x y w h	x-position y-postion width height	Creates a new dropdown at the given position with the given style. Height is defined in items. Dropdowns can be filled with items using *Cadditem*. To access certain items you can use *Csetcursel* and *Cgetcontent*.
Cpopup(x,y,w,h)		
x y w h	x-position y-postion width height	Creates a new Popup at the given position with the given style. Height is defined in items. Popups can be filled with items using *Cadditem*. To access certain items you can use *Csetcursel* and *Cgetcontent*.Popups are not triggered by an event in the main event-handler, but by an event-action of another control.
Cmenu(x,w,wt)		
x w wt	x-position topic width menu width	Creates a new menu at the given position with the given style. Topic width defines the width which will be marked black when clicking on the menu topic, menu width defines the width of the dropdown containing the menu items. Menus can be filled with item using *Cadditem*. To access certain items you can use *Csetcursel* and *Cgetcontent*. Menus can be added to the main menu bar using *Caddmenu*.
Cmenubar()		
		Creates the main menubar at the given position with the given style. Menus are added using *Caddmenu*. You can only define one menubar.

Cbutton(x,y,w,h,l,b)		
x y w h l b	x-position y-postion width height line: 0 = white 1 = black 2 = gray border: curves of edges	Creates a new button at the given position with the given style. Buttons are labeled with text using *Csetcontent*.
Cimage(x,y,w,h)		
x y w h	x-position y-postion width height	Creates a new clickable image at the given position with the given style. The image shown by the control is defined by using *Csetcontent* with the picture resource as string parameter. Please take care to use the same width and height as the picture when defining the Cimage control.
Cradio(x,y,w,h)		
x y w h	x-position y-postion width height	Creates a new radio control at the given position with the given style. Radios can be filled with items using *Cadditem*. To access certain items you can use *Csetcursel* and *Cgetcontent*.
Cswitch(x,y,w,h,l,b)		
x y w h l b	x-position y-postion width height line: 0 = white 1 = black 2 = gray border: curves of edges	Creates a new switch control at the given position with the given style. To access the state of the control you can use *Csetstate* and *Cgetstate*.
Ccheckbox(x,y,w)		
x y w	x-position y-postion width	Creates a new checkbox at the given position with the given style. To access the state of the control you can use *Csetstate* and *Cgetstate*.
Ctable(x,y,w,h,r)		
x y w h r	x-position y-postion width height number of rows	Creates a new table at the given position with the given style. Tables can be filled with items using *Cadditem* (to add a new line and the first field item) in combination with *Csetfield*.(to fill any field items) To access certain items you can use *Cgetfield* (or *Cgetcontent* for the first field item). Sizes and styles of the rows are determined by using *Csetrow*.
CtableDB(x,y,w,h,r,pDB,sF)		
x y w h r pDB sF	x-position y-postion width height number of rows database handle string with access format	Creates a new table at the given position with the given style. This kind of table is not filled using *Cadditem* but by directly accessing a PcoketC database using a database handle. It will display the first r fields of the database where sF is the formatter string for the database access (compare to the PocketC function *dbreadx*). You are not able to add new items using *Cadditem* or to change it with *Csetfield* - you have to change contents using the standard PocketC database functions. All other access functions are the same as for the standard table control. Examine the code example to get more information on how to use database

		driven tables.
Cframe(x,y,w,h)		
x y w h	x-position y-postion width height	Creates a new dialog frame at the given position with the given style. In order to place controls on the dialog correctly you have to create the frame as the first control (otherwise it would overlap other controls). Dialog frames usually have their own event handler (examine the code example to get more information).
Cdestroy(Ch)		
Ch	handle of control	Destroys a control and frees the memory which was allocated for the control (and ist items).

Drawing functions

Cdraw(Ch)		
Ch	handle of control	Draws (or re-draws) a control. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Ctable, CtableDB, Cdropdown, Clistbox, Clabel
Cerase(Ch)		
Ch	handle of control	Erases a control from the screen (without destroying it). The control can be re-drawn using *Cdraw*. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Ctable, CtableDB, Cdropdown, Clistbox, Clabel
Chide(Ch)		
Ch	handle of control	Hides a control. It can be show again using *Cshow*. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Ctable, CtableDB, Cdropdown, Clistbox, Clabel
Cshow(Ch)		
Ch	handle of control	Shows a control which was hidden using *Chide*. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Ctable, CtableDB, Cdropdown, Clistbox, Clabel
Cdeactivate(Ch)		
Ch	handle of control	Deactivates the functionality of the control (without hiding it). Some controls (e.g. buttons) change their appearance when deactivated. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox, Clabel
Cactivate(Ch)		
Ch	handle of control	Re-activates a control which was deactivated using *Cdeactivate*. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox, Clabel

Access functions

Csetrow(Ch,r,w,a)		
Ch r w a	handle of control row index width of row alignement: 0 = left 1 = center 2 = right	Determines the layout of table rows. Please keep in mind that row contents are NOT automatically limited by row widths. That means that if row contents are too long this will result in "jerky" table layouts. to be used with: Ctable, CtableDB
Csetstate(Ch,s)		
Ch s	handle of control state: 0 = inactive 1 = active	Sets the state of the control. to be used with: Cswitch, Ccheckbox
Cgetstate(Ch)		
Ch	handle of control	Gets the state of the control. Returns 0 or 1. to be used with: Cswitch, Ccheckbox
Csetcursel(Ch,i)		
Ch i	handle of control index of item	Sets the selected item. The first item has an index of 0. By giving an index value of -1 no item is selcted. After setting the index the content of items can be read by using *Cgetcontent* or *Cgetfield*. it can be changed by using *Csetcontent* or *Csetfield*. to be used with: Cradio, Cmenu, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox
Cgetcursel(Ch)		
Ch	handle of control	Gets the index of the currently selected item. The first item has the index 0. Returns -1 if no item is selected. to be used with: Cradio, Cmenu, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox
Csettopic(Ch,s)		
Ch s	handle of control content string	Sets the topic text of the control. For menus this is the text shown in the menubar, for edit fields this is the text shown at the top of the dialog prompting when the control is clicked for new input (if no topic is set, the current content of the control will be displayed instead). to be used with: Cedit, Cmenu
Cgettopic(Ch)		
Ch	handle of control	Gets the topic text of the controls. Returns a string value. to be used with: Cedit, Cmenu
Csetfield(Ch,r,s)		
Ch r s	handle of control number of row content string	Sets the content of a table field. The first field has the index number 0. The first field can be set while using *Cadditem* to add a new line, all other fields must be filled using *Csetfield*. Please keep in mind that the length of the content strings is not controlled by the control automatically, so that long strings can result in "jerky" table

		layouts. to be used with: Ctable, CtableDB
Cgetfield(Ch,r)		
Ch r	handle of control number of row	Gets the content of a table field. Returns a string vale. The first field has the index number 0. The content of the first field can also be read by using *Cgetcontent*. to be used with: Ctable, CtableDB
Csetcontent(Ch,s)		
Ch s	handle of control content string	Sets the content of the control. Please keep in mind that the length of the content strings is not controlled by the control automatically, so that long strings can result in "jerky" control layouts. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Cmenu, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox, Clabel
Cgetcontent(Ch)		
Ch	handle of control	Gets the current content of the control. Returns a string value. When used for controls with more than one item, please call *Csetcursel* first to select the item. When used for tables, only the first field of a row will be read. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cradio, Cmenubar, Cmenu, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox, Clabel
Csetsize(Ch,n)		
Ch n	handle of control number of items	Allocates the memory for a certain number of items for the control. If a control has a large number of items, this method is faster than only calling *Cadditem.* to be used with: Ctable, CtableDB, Cdropdown, Cpopup, Clistbox
Cadditem(Ch,s)		
Ch s	handle of control item string	Adds a new item to the control. Items will be added at the end of the item list. Automatically increaeses the allocated memory by one if the pre-allocated memory (by using *Csetsize*) is not sufficient. To change existing items please use *Csetcontent* in combination with *Csetcursel*. to be used with: Ccheckbox, Cradio, Cmenu, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox
Cremoveitem(Ch)		
Ch	handle of control	Removes an item. To select a certain item, please use *Csetcursel*. The allocated memory for the item list will automatically be decreased by one after the item was deleted. to be used with: Ccheckbox, Cradio, Cmenu, Ctable, CtableDB, Cdropdown, Cpopup, Clistbox
Caddmenu(Ch,Cm)		
Ch	handle of control	Adds a menu to the main menubar. Please keep in mind that it is not

Cm	handle of menubar	possible to change the order of existing menus, so you have to add them in the right order when calling *Caddmenu*. There is no command to remove menus. If you want to do this, please use *Cdestroy* for all menus and the menubar and then re-define them. to be used with: Cmenubar

Event functions

Cevent(Ch,e)		
Ch e	handle of control event	Checks if the given control was activated. The event *e* is trapped by the PocketC *event(1)* function. To survey all GUI elements, you have to use a message loop which catches an event and then performs *Cevent* for all controls (please examine the example code for more information). Returns 1 if the control was activated, otherwise 0. To determine actions of popup controls *Cpopupevent* has to be used. to be used with: Cbutton, Cimage, Cswitch, Cedit, Ccheckbox, Cmenubar, Cradio, Ctable, CtableDB, Cdropdown, Clistbox, Clabel

Cpopupevent(Ch)		
Ch	handle of control	Opens the popup list and checks if an item was selected by the user. Returns 1 if an item was selected, otherwise 0. *Cpopupevent* is usually be called in response to an event of another control. to be used with: Cpopup

CEditor Usage

To produce an automatically generated PocketC application you have to perform the following steps:

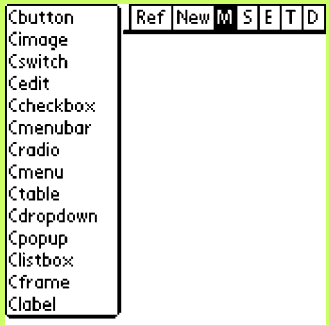
- design your GUI by placing the appropriate controls
- design your menubar by adding the menus
- customize the properties of your controls if necessary (e.g. modify the control variables)
- save your form to be able to perform later modifications
- let CEditor generate the application code for you
- modify the "on_myvariable()"-functions in the code to perform your own control actions

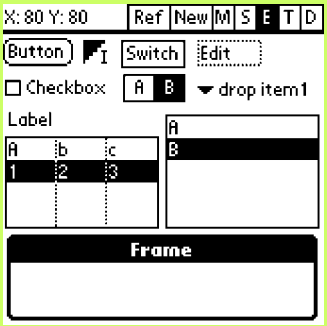
CEditor generates the following code-memos for you:

- // MyApplication main
  Main application frame. Initializes the graphic output and calls all main functions. Includes the CControls function library codes.
- /$ MyApplication controls
  Initializes all controls and its contents and draws them.
- /$ MyApplication methods
  Handles all control events in a message loop and offers the "on_myvariable()"-functions for each control.

CEditor functions

Screen buttons

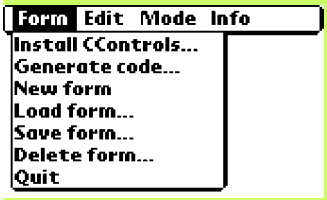




Ref: refresh the screen (sometimes necessary to avoid "junkiness")
New: create a new control (will be followed by the property dialog)
M: move controls
S: change the size of controls
E: edit controls (choose one to get the property dialog)
T: test controls (to check all functions of your GUI)
D: delete controls

To access controls, which are normally hidden (like popups and menus) you can use special menu commands.

Form menu

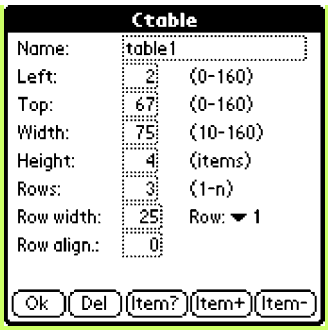


Install CControls: generates three code-memos with the CControls function library
Generate code: generates the code-memos for your current form
New form: create a new empty form
Load form: loads a formerly saved form-project
Save form: saves the current form in a form-project
Delete form: deletes a formerly saved form-project

Edit menu

New control: inserts a new control into your form
Refresh screen: redraws the form (sometimes necessary)
Edit popup: lets you choose a popup-control for editing
Edit menu: lets you choose a menu for editing
Edit form: go back to form after editing popups or menus

Control properties (example)



All properties of the supported controls can be modified in the corresponding control property dialogs. Here you can find the same attributes which are used when modifying controls with the CControls library functions. Controls whch support the "Cadditem"-function have additional "item"-buttons, tables have an additional dropdown to choose the row to be modified by the current "row"-attributes.
Please refer to the CControls function reference to get more descriptions on the controls attributes.
The "name"-attribute determines the name of Chandle-type variable to be used for the control when generating the code-memos.

Aknowledgements and legal stuff

CControls and CEditor can be used for free. I am a freelancing consultant and these programs are part of my "active advertising".

The only "price" for the usage of CControls is that you may mail me a copy of an interesting application which you have programmed using CControls (it would be great to see what can be done with PocketC and CControls).

This software is provided as is, with no guarantee of fitness for any particular task. The user assumes all responsibility for its use.

If you have any comments please feel free to contact me: mail@mscon.de / www.mscon.de