Plumber Custom Control Pack - 1

Support

Support

Registered users can expect full support through one of the methods listed below. Like many of you, I've suffered emotional trauma from a lukewarm product with marginal support. While I won't get on a soapbox and dictate my ideas on the wrongs poor support creates, I think it is sufficient to say a company should not sell a product if they can't support it. Even the best of us will need help once in a while. Paraphrasing from a recent television commercial, "If I don't return your phone call (Email in this case), it's likely I'm dead."

I check each of these online services at least daily and will make every reasonable attempt to answer all questions without delay. Support through the mail system is acceptable also, although much slower.

Compuserve	71101,1063
AOL	rpg3
Internet	rpg@metronet.com
The Mail	Plumber Programming P.O. Box 914 Bedford, TX 76095

Feedback

To make these controls truly useful, I need to know what you do and what you need. Otherwise, the controls will only contain features I find useful in my applications. Any comments, suggestions, critiques, and ideas are very welcome. Please send any of it to one of the address listed above.

Custom Controls

I'm always looking for future projects. If you have an idea or need a custom control written, give me some information on what you need or want.

Registration

Registration

When you register the Plumber Control Pack 1, you'll receive a diskette containing the complete source code for all of the controls as well as the source code for the demonstration program. The registration cost for the controls is $75.

Compuserve

If you are a member of CompuServe, you can easily register these controls by typing, "GO SWREG" and following the directions there. The registration ID number for the Plumber Control Pack 1 is 2595.

The Mail

You can also register this program by sending $75 to the address listed in the support section. For your convenience, this help file contains an order form that can be printed and mailed in.

Order Form

Plumber Custom Control Pack - 1

To print this form, select `Print Topic` from the `File` menu.

Name:	____________________
Company:	____________________
Address:	____________________
	____________________
City:	____________________
State:	________ Zip Code: ____________ Country: ______________
Phone:	____________________
Fax:	____________________
EMail:	____________________

Diskette: [ ] 3½ [ ] 5¼

Quantity: [ ]	$75	each	______________
Texas Sales Tax:	$ 5.44	each	______________
International Shipping & Handling (Except: Canada and Mexico)	$ 5		______________
Total Payment:			______________

Enclose your check payable to Plumber Programming

Send to: Plumber Programming
P.O. Box 914
Bedford, TX 76095

Plumber Custom Control Pack - 1

Plumber Custom Control Pack - 1

Plumber Custom Control Pack - 1

Plumber Custom Control Pack - 1

FauxCmb

⬇ The FauxCmb control provides a false combo box button that can be placed in a Dialog, Form View, or anywhere that a button can be created.

FauxCmb Messages

If you want to handle messages send by the FauxCmb control to its associate window, add a message-map entry and message-handler member function to the associate window class for each message.

By default, the control's associate window is its parent window, determined during creation. The associate window can be easily changed by using SetAssociate.

Each message-map entry takes the following form.

ON_MESSAGE (*PPFC_Message*, *memberFxn*)

where *PPFC_Message* is the message you want to respond to and *memberFxn* is the name of the associate window member function you have written to handle the message.

The associate window member function has the form:

```
afx_msg LRESULT memberFxn (WPARAM WParam, LPARAM LParam);
```

Possible messages sent by the FauxCmb control are:

Message	Sent to associate window when...
PPFC_BTNCLICKED	The user clicks on the button
PPFC_ASSOCIATELOSS	The window has lost its association with the control
PPFC_ASSOCIATEGAIN	The window has gained an association with the control

FauxCmb Construction
#include <FauxCmb.h>

CFauxCmb::CFauxCmb

CFauxCmb();

Remarks Constructs a CFauxCmb object. When creating a CFauxCmb object within your code, first call the CFauxCmb constructor to construct the CFauxCmb object; then call the Create member function to create the control and attach it to the CFauxCmb object.

CFauxCmb is derived from CWnd.

Note: FauxCmb makes use of the OEM bitmap OBM_COMBO. In order for this resource to be available, you must define the constant OEMRESOURCE before including WINDOWS.H. If you are using precompiled headers, place the following statement in STDAFX.H before any #include statements:

```
#define OEMRESOURCE
```

See Also Creating the controls within App Studio

FauxCmb Operations
#include <FauxCmb.h>

CFauxCmb::Create

BOOL Create (DWORD *dwStyle*, **CRect&** *Rect*, **CWnd*** *pParent,* **UINT** *nID)*;

dwStyle - Specifies the control's style.

Rect - Specifies the control's size and position. See the remarks, below for more information.

pParent - Specifies the control's parent window, usually a CDialog.

nID - Specified the control's ID.

Remarks You create a CFauxCmb object in two steps. First call the constructor, then call the Create function.

The CRect object parameter specifies the control's position and size. If the CRect has a zero width or zero height, the control is resized automatically to the same width or height as a standard combo box button (reference the source code OnCreate function for further information).

Apply the following window styles to the FauxCmb control:

WS_CHILD	- Always
WS_VISIBLE	- Usually
WS_DISABLED	- Rarely
WS_GROUP	- To group controls
WS_TABSTOP	- To include the button in the tabbing order

Return Value Nonzero if successful; otherwise 0

CFauxCmb::GetAssociate

CWnd* GetAssociate ();

Remarks Retrieves the current CWnd object associated with the control. All messages are sent from the control to the associate window.

CFauxCmb::SetAssociate

CWnd* SetAssociate (CWnd* *pNewWnd***);**

pNewWnd - Specified the new CWnd object to associate with the control.

Remarks Sets the CWnd object associated with the control. All messages are sent from the control to the associate window.

Return Value A pointer to the previous associate CWnd object.

DiagScroll

The DiagScroll control provides a diagonal scroll bar or spinner button that can be placed in a Dialog, Form View, or anywhere that a button can be created.

DiagScroll Construction

#include <DiagScrl.h>

CDiagScroll::CDiagScroll

CDiagScroll();

Remarks Constructs a CDiagScroll object. When creating a CDiagScroll object within your code, first call the CDiagScroll constructor to construct the CDiagScroll object; then call the Create member function to create the control and attach it to the CDiagScroll object.

CDiagScroll is derived from CWnd.

See Also Creating the controls within App Studio

Calendar Messages

If you want to handle messages send by the Calendar control to its associate window, add a message-map entry and message-handler member function to the associate window class for each message.

By default, the control's associate window is its parent window, determined during creation. The associate window can be easily changed by using SetAssociate.

Each message-map entry takes the following form.

ON_MESSAGE (*PPC_Message*, *memberFxn*)

where *PPC_Message* is the message you want to respond to and *memberFxn* is the name of the associate window member function you have written to handle the message.

The associate window member function has the form:

```
afx_msg LRESULT memberFxn (WPARAM WParam, LPARAM LParam);
```

Possible messages sent by the Calendar control are:

Message	Sent to associate window when...
PPC_ASSOCIATELOSS	The window has lost its association with the control
PPC_ASSOCIATEGAIN	The window has gained an association with the control
PPC_DATECHANGED	Sent when the selected date has been changed through any manner: keyboard, mouse, or program.
PPC_DATEFINISHED	Sent for any of the following reasons: 1. The user pressed Enter to select a date 2. When the user has clicked (& released) on a valid date, if the control is created with the PPC_SINGLECLICK style. 3. The control lost its input focus, if the control is created with the PPC_HIDELOSEFOCUS style.

DiagScroll Messages

If you want to handle messages send by the DiagScroll control to its associate window, add a message-map entry and message-handler member function to the associate window class for each message.

By default, the control's associate window is its parent window, determined during creation. The associate window can be easily changed by using SetAssociate.

Each message-map entry takes the following form.

```
ON_MESSAGE (PPDS_Message, memberFxn)
```

where *PPDS_Message* is the message you want to respond to and *memberFxn* is the name of the associate window member function you have written to handle the message.

The associate window member function has the form:

```
afx_msg LRESULT memberFxn (WPARAM WParam, LPARAM LParam);
```

Possible messages sent by the DiagScroll control are:

Message	Sent to associate window when...
PPDS_ASSOCIATELOSS	The window has lost its association with the control
PPDS_ASSOCIATEGAIN	The window has gained an association with the control

When the DiagScroll control is clicked on, it sends a WM_VSCROLL or WM_HSCROLL message, depending on how the control was created. To respond to these messages, use the Class Wizard to add the appropriate member function. To do this manually, add one of the following to the associate window message-map:

```
ON_WM_VSCROLL()
ON_WM_HSCROLL()
```

Along with that, add one of the following member functions:

```
afx_msg void OnVScroll (UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
afx_msg void OnHScroll (UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
```

DiagScroll Operations

#include <DiagScrl.h>

CDiagScroll::Create

BOOL Create (DWORD *dwStyle*, **CRect&** *Rect*, **CWnd*** *pParent,* **UINT** *nID)*;

dwStyle - Specifies the control's style.

Rect - Specifies the control's size and position.

pParent - Specifies the control's parent window, usually a CDialog.

nID - Specified the control's ID.

Remarks You create a CDiagScroll object in two steps. First call the constructor, then call the Create function.

Apply the following window styles to the DiagScroll control:

WS_CHILD	- Always
WS_VISIBLE	- Usually
WS_DISABLED	- Rarely
WS_GROUP	- To group controls
WS_TABSTOP	- To include the control in the tabbing order
PPDS_VERTICAL	- Control has vertical arrows and sends WM_VSCROLL messages
PPDS_HORIZONTAL	- Control has horizontal arrows and sends WM_HSCROLL messages
PPDS_AUTOSIZE	- Control's width is automatically sized based upon its height
PPDS_DEFAULT	- a combination of the WS_BORDER, PPDS_VERTICAL, and PPDS_AUTOSIZE styles

Return Value Nonzero if successful; otherwise 0

CFauxCmb::GetAssociate

CWnd* GetAssociate ();

Remarks Retrieves the current CWnd object associated with the control. All messages are sent from the control to the associate window.

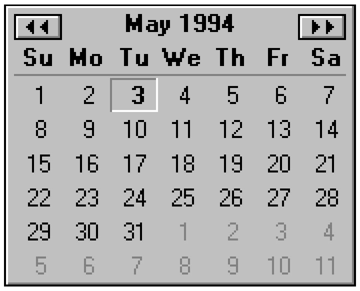
CFauxCmb::SetAssociate

CWnd* SetAssociate (CWnd* *pNewWnd***);**

pNewWnd - Specified the new CWnd object to associate with the control.

Remarks	Sets the CWnd object associated with the control. All messages are sent from the control to the associate window.
Return Value	A pointer to the previous associate CWnd object.

Calendar


May 1994

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

The Calendar control provides a convenient, easy to use calendar from which a user can select or view a date. The buttons and keyboard can be enabled and disabled. Dates can be marked as well as selected. The control can be set to use any available font, and can be automatically resized to fit the selected font. It can be placed in a Dialog, Form View, or anywhere that a button can be created. Used in conjunction with the EditDate control, it provides an interface similar to a combo box for selecting a date visually.

Calendar Construction

#include <Calendar.h>

CCalendar::CCalendar

CCalendar();

Remarks Constructs a CCalendar object. When creating a CCalendar object within your code, first call the CCalendar constructor to construct the CCalendar object; then call the Create member function to create the control and attach it to the CCalendar object.

CCalendar is derived from CWnd.

See Also Creating the controls within App Studio

Calendar Operations
#include <Calendar.h>

CCalendar::ClearMarked

void ClearMarked (BOOL *Update* = FALSE);

Update - TRUE to update the control immediately after clearing any marked dates.

Remarks Clears any marked dates in the current calendar shown.

CCalendar::Create

BOOL Create (DWORD *dwStyle*, **CRect&** *Rect*, **CWnd*** *pParent,* **UINT** *nID)*;

dwStyle - Specifies the control's style.

Rect - Specifies the control's size and position.

pParent - Specifies the control's parent window, usually a CDialog.

nID - Specified the control's ID.

Remarks You create a CDiagScroll object in two steps. First call the constructor, then call the Create function.

Apply the following window styles to the DiagScroll control:

WS_CHILD - Always
WS_VISIBLE - Usually
WS_DISABLED - Rarely
WS_GROUP - To group controls
WS_TABSTOP - To include the control in the tabbing order

PPC_SHOWENDS - Calendar will show prior/next month dates
PPC_SELECTENDS - Allow user to select prior/next month dates.
PPC_SHOWSELECTED - Show the selected date with a surrounding box.
PPC_ENABLEBUTTONS - Enable/show the buttons on the calendar
PPC_ENABLEKEYBOARD - Enable keyboard keys described in Using the Calendar.
PPC_SINGLECLICK - A single click on a date will cause the PPC_DATEFINISHED message to be sent.
PPC_HIDELOSEFOCUS - The calendar will be hidden when it loses its input focus.
PPC_AUTOSIZE - The calendar is automatically sized based upon the font used.
PPC_DEFAULT - a combination of the WS_BORDER, PPC_SHOWSELECTED, PPC_ENABLEBUTTONS, PPC_ENABLEKEYBOARD, PPC_SINGLECLICK, and PPC_AUTOSIZE styles

Return Value Nonzero if successful; otherwise 0

CCalendar::GetAssociate

CWnd* GetAssociate ();

Remarks Retrieves the current CWnd object associated with the control. All messages are sent from the control to the associate window.

CCalendar::GetAutoSize

BOOL GetAutoSize();

Return Value Returns TRUE if the control was created with the PPC_AUTOSIZE style.

CCalendar::GetControlHeight

int GetControlHeight();

Return Value Returns a the height of the control. If the control window has been created (using Create), it returns the actual height of the control. If the control window has not been created, it returns the suggested height of the control based on the currently selected font.

CCalendar::GetControlWidth

int GetControlWidth();

Return Value Returns a the width of the control. If the control window has been created (using Create), it returns the actual width of the control. If the control window has not been created, it returns the suggested width of the control based on the currently selected font.

CCalendar::GetButtonsActive

BOOL GetButtonsActive();

Return Value Returns TRUE if the control was created with the PPC_ENABLEBUTTONS style.

CCalendar::GetKeyboardActive

BOOL GetKeyboardActive();

Return Value Returns TRUE if the control was created with the PPC_ENABLEKEYBOARD style.

CCalendar::GetSelectEnds

BOOL GetSelectEnds();

Return Value Returns TRUE if the control was created with the PPC_SELECTENDS style.

CCalendar::GetShowEnds

BOOL GetShowEnds();

Return Value Returns TRUE if the control was created with the PPC_SHOWENDS style.

CCalendar::GetShowSelected

BOOL GetShowSelected();

Return Value Returns TRUE if the control was created with the PPC_SHOWSELECTED style.

CCalendar::GetSingleClickOk

BOOL GetSingleClickOk();

Return Value Returns TRUE if the control was created with the PPC_SINGLECLICK style.

CCalendar::GetSelectedDate

CTime GetSelectedDate();

Return Value Returns a CTime object representing the currently selected date.

CCalendar::MarkDate

BOOL MarkDate (CTime *Date*, **BOOL** *Mark* = TRUE, **BOOL** *Update* = TRUE);

Date	- Specifies the date to mark/unmark
Mark	- TRUE to mark the date, FALSE to clear any marks.
Update	- TRUE to update the control immediately after marking the date.

Remarks	If you have several dates to mark/unmark, call this function with a FALSE value for *Update.* On the last call to MarkDate, call this function with a TRUE value for *Update.*
Return Value	Returns TRUE if the *Date* exists in the calendar currently shown.

CCalendar::SelectDate

BOOL SelectDate (CTime *Date*);

Date - Specifies the date to select

Remarks	Selects the *Date* in the calendar. If the date does not exist in the calendar currently shown, the calendar's month is changed to include the Date.
Return Value	Returns TRUE if the *Date* exists in the calendar currently shown.

CCalendar::SetAssociate

CWnd* SetAssociate (CWnd* *pNewWnd***);**

pNewWnd - Specifies the new CWnd object to associate with the control.

Remarks	Sets the CWnd object associated with the control. All messages are sent from the control to the associate window.
Return Value	A pointer to the previous associate CWnd object.

CCalendar::SetTextFont

void SetTextFont (CFont* *pFont*, **BOOL** *Resize* = TRUE);

pFont	- Specifies the new CFont to use when painting the calendar.
Resize	- TRUE to automatically resize the control based on the font.

Remarks Sets the font used when painting the calendar control. If *Resize* is FALSE, the control is not sized to accommodate the font.

Note: The Calendar control's text font can also be set using CWnd::SetFont, however, the control will not be resized when changing its font in this manner.

CCalendar::SetTextColor	- The basic text color
CCalendar::SetEndsTextColor	- The color for the prior/next month's dates
CCalendar::SetMarkedTextColor	- The marked dates text color
CCalendar::SetSelectedTextColor	- The selected date's text color

CCalendar::SetBackColor - The basic text background color
CCalendar::SetEndsBackColor - The background text color for the prior/next month's dates
CCalendar::SetMarkedBackColor - The marked dates background text color
CCalendar::SetSelectedBackColor - The selected date's background text color

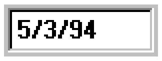
COLORREF SetTextColor (COLORREF *Color*);

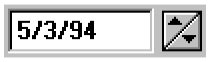
Color	- Specifies the new color to use when painting the specified element.

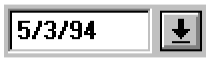
Remarks Each of the eight functions above sets various color aspects of the calendar control.

Return Value Returns a COLORREF relating the the old color for the specified element.

EditDate







The EditDate control provides a number of simplified methods of date input. Dates can be entered and parsed in a variety of formats, they can be easily incremented with a convenient diagonal scroll box, or they can easily be selected visually from a drop down calendar*.

*Similar in function to a well known home-finance program.

EditDate Construction

#include <EditDate.h>

CEditDate::CEditDate

CEditDate();

Remarks Constructs a CEditDate object. When creating a CEditDate object within your code, first call the CEditDate constructor to construct the CEditDate object; then call the Create member function to create the control and attach it to the CEditDate object.

CEditDate is derived from CEdit.

Note: EditDate uses the DiagScroll, FauxCmb, and Calendar custom controls. When using EditDate, be sure to include these other control's .CPP files in your project.

EditDate Operations
#include <EditDate.h>

CEditDate::AttachCombo

void AttachCombo ();

Remarks Creates and attaches a FauxCmb object to the edit control, enabling the ability to select a date visually from a drop-down calendar.

CEditDate::AttachScroll

void AttachScroll ();

Remarks Creates and attaches a DiagScroll object to the edit control, enabling the ability to scroll through a range of dates easily.

CEditDate::DetachCombo

void DetachCombo ();

Remarks Destroys and removes a previously attached FauxCmb object.

CEditDate::DetachScroll

void DetachScroll ();

Remarks Destroys and removes a previously attached DiagScroll object.

CCalendar::GetDate

CTime GetDate ();

Remarks This function parses the text in the edit control to determine the date, using the current format. If the date in the edit control is invalid, the date is set to the current date using the system clock.

Return Value Returns a CTime object containing the date displayed.

CCalendar::GetDateStr

void GetDateStr (CString& *StrDate***);**

StrDate - on return, contains the date entered in the edit control, validated against the current date format.

Remarks This function parses the text in the edit control to determine the date, using the current format. If the date in the edit control is invalid, the date is set to the current date using the system clock.

CCalendar::InitDate

void InitDate (CString& *StrDate,* **DateFormat** *Format,* **BOOL** *InitIfBlank* = TRUE**);**

StrDate - on return, contains the date entered in the edit control, validated against the current date format.

Format - Desired date format. This can be one of the following

MMDDYY	- 1/24/94
DDMMM	- 23JAN
DDMMMYY	- 23JAN94
MMMDDYY	- JAN2394

InitIfBlank - If TRUE, the date is set to that parsed from *StrDate*.
- If FALSE, the date is set to that parsed from *StrDate* only when *StrDate* is not blank. If *StrDate* is blank, the date is set to the current date using the system clock.

Remarks This function parses the *StrDate* parameter to determine the date, using the specified format. If the *StrDate* value is invalid, the date is set to the current date using the system clock.

CCalendar::ShowCalendar()

void ShowCalendar ();

Remarks Causes the drop-down calendar to be displayed, allowing visual selection of a date.

Creating the controls within App Studio

Note: App Studio does not allow the visual editing of controls other than VBX controls. Because the Plumber Custom Controls are derived from MFC code, you will not be able to visually edit and preview them. With the App Studio, you can easily define a control's placement and window styles.

To create one of these controls within App Studio, follow the following steps:

1. Register the control. Anywhere before the control is first created, use the **RegisterControl** function to register the control's window class with windows. A convenient place to do this is within your applications **InitInstance** function. A list of each of the necessary #include files and RegisterControl commands is listed below. As an example:

```
#include <FauxDlg.h>    // include the necessary file

BOOL CMyApp::InitInstance ()
{
      //----- Other initialization ...

      CFauxDlg::RegisterControl();
      // Register the control
}
```

2**. Use App Studio to create a user control.** The user control is depicted in the control palette as a person's head looking to the right.

a. Set the **class** name to the appropriate class name listed below.

b. Set the **style** to the appropriate value. You'll have to manually figure out what the style value should be. Reference windows.h as well as the control's .h file for the values of each desired style, and add them together. The Visible, Group, Tabstop, and Disabled styles can be set with the control properties dialog box, but everything else needs to be set manually.

Control	**Include File**	**Class Name**	**RegisterControl**
Calendar	CalenDlg.h	PPCalendar	CCalenDlg::RegisterControl();
DiagScroll	DiagDlg.h	PPDiagScroll	CDiagDlg::RegisterControl();
FauxCmb	FauxDlg.h	PPFauxCombo	CFauxDlg::RegisterControl():

Description

The Plumber Custom Control Packs provide you -- the programmer -- with a series of custom controls unlike any other.

The Plumber Custom Control Packs are unique because:

1. **The controls are coded completely in C++ using MFC.**

2. **The controls are small, fast, and efficient.**

3. **The controls are reasonably priced.**

4. **The controls are included with complete source code.**

The controls in this control pack include:

1. A diagonal scroll or "spinner" button.

2. A false combo box button that can be placed anywhere.

3. A calendar control allowing easy visual selection of a date.

4. A formatted date edit control that supports four different date formats and several methods of selecting a date including a drop-down calendar.

The Plumber Custom Control Packs are not:

1. Just another set of controls like the rest or everything else out there.

2. Kitchen sinks containing one of everything with every possible option (and more). The controls have been selected to provide commonly needed functionality without carrying a lot of overhead. Speed, size, and power are important after all!

3. They aren't VBX controls! VBX controls will never be supported in a 32 bit environment and will become just another passing fad when the next version of Windows, Chicago, is released. (And I won't go on about the clumsy interface you have to go through to use VBX controls.)

4 Finally, they controls aren't supplied by a large autonomous corporation only looking after the bottom line. Plumber Programming is a small two person partnership, interested in the bottom line as well, but bright enough to realize the road to success is to make and keep its customers (You!) happy.

Using the Calendar

When the Calendar control has the input focus, the keys listed below have the action specified:

H	Sets the selected date to the last day of the montH.
K	Sets the selected date to the last day of the weeK.
M	Sets the selected date to the first day of the Month.
R	Sets the selected date to the last day of the yeaR, Dec 31.
T	Sets the selected date to Today's date, based upon the system clock.
W	Sets the selected date to the first day of the Week.
Y	Sets the selected date to the first day of the Year, Jan 1.

-	Sets the selected date to that one day back.
+	Sets the selected date to that one day forward.

Home	Sets the selected date to the first day of the year, Jan1.
End	Sets the selected date to the last day of the year, Dec 31.

PgUp	Sets the selected date to that one month back.
PgDn	Sets the selected date to that one month forward.

←	Sets the selected date to that one day back.
→	Sets the selected date to that one day forward.
	Sets the selected date to that one week back.
↓	Sets the selected date to that one week forward.

Enter	Causes a PPC_DATEFINISHED message to be sent.

Using the EditDate Control

When the EditDate control has the input focus, the keys listed below have the action specified:

When using all date formats:

-	Sets the selected date to that one day back.
+	Sets the selected date to that one day forward.

When using the MMDDYY date format:

H	Sets the selected date to the last day of the montH.
K	Sets the selected date to the last day of the weeK.
M	Sets the selected date to the first day of the Month.
R	Sets the selected date to the last day of the yeaR, Dec 31.
T	Sets the selected date to Today's date, based upon the system clock.
W	Sets the selected date to the first day of the Week.
Y	Sets the selected date to the first day of the Year, Jan 1.