Article

Browse Code

Stats

Revisions

Alternatives

Comments & Discussions (11)

Add your own alternative version

Control Arrays on a Dialog box/FormView

By **Alton Williams**, 21 Sep 2005

★ ★ ★ ★ ★ 4.43 (10 votes)

Download Dialog based demo project - 19.70 Kb

Download FormView SDI demo project - 29.4 Kb

About Article

Using control arrays in VC++.

Type	**Article**
Licence	
First Posted	**1 Mar 2005**
Views	**57,379**
Bookmarked	**25 times**

VC6 Dev Intermediate

Bookmark Print Email

Table of contents

Introduction

I have received so much help from *The Code Project* community. I felt a contribution from me is a way of saying thanks to all. Here's a modified version of the project to cater for FormView apps as well. Many thanks to Joe for pointing it out.

Visual Basic has the facility to handle Control Arrays.

The advantages of using them when you have multiple child windows of the same control are as follows:

1. Less lines of code when you want to modify or inspect properties of the controls (i.e. loops).
2. You wish to do (1. above) to some of them (i.e. the first or last five, every other or third).
3. When you wish to know which slider or spin control was scrolled when Windows traps it.
4. The ease to directly modify other controls by index (without using `CWnd::UpdateData()`).
5. Pointer arithmetic and comparison becomes available (cutting the use of `CWnd::GetDlgItem()`).

As I said it can be done in VB and there's no reason why you can't in VC++. You can! It's just done differently. This tutorial shows an example of how to make use of it.

Getting started

Let's start at the most appropriate place, the beginning. Create an MFC EXE app. For simplicity I'll use a Dialog based one (if you select a single or multiple document, follow the AppWizard through to step 6 of 6 Base Class (lower left) and select `CFormView`).

Having done that, we now have the resource editor.

We are in a position to place the controls on the dialog (or FormView).

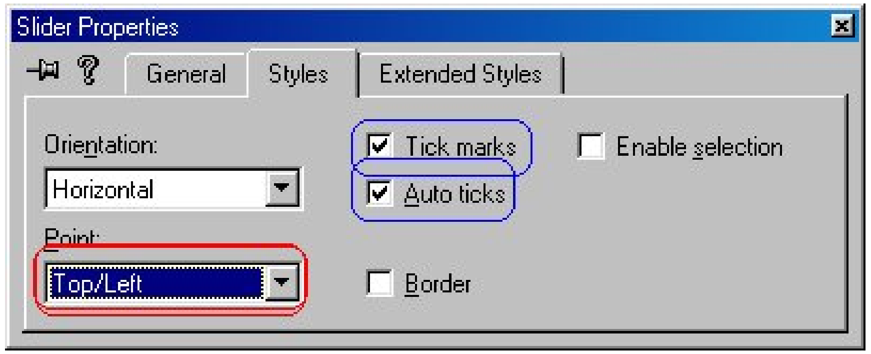
The controls that we require for this demo are each of the following:

1. Edit Box
2. Spin Button control
3. Slider control

Select them one by one and place them accordingly. Having placed them on the dialog, we have to set the sliders to have tick marks. So select the slider only and then right click the mouse. Then select "Properties".

Top News

On the Slider Properties select the "Styles" tab. Check both "Tick marks" and "Auto Ticks". Also for point, select "Top/Left". Then close. This is done so you don't have to edit the properties for the other two.


Slider Properties

General | Styles | Extended Styles

Orientation: Horizontal

Point: Top/Left

☑ Tick marks

☑ Auto ticks

☐ Enable selection

☐ Border

Then holding down the Shift key select the other two controls. Copy (CTRL-C) and paste (CTRL-V). Move the pasted controls somewhere else and repeat the paste. Align them so that they look presentable.

The completed dialog:

Related Articles

Enumerate Controls In a Dialog Box or FormView

A Custom Dialog Box Control

Dialog Box Control Management

Layout Manager for Dialogs, Formviews, DialogBars and PropertyPages

Taming the FormView: Understanding how to Implement the FormView Control Effectively

How to enable context sensitive help for individual controls on a FormView

How to Browse Network Folders using Folder Dialog Box Control ?

Tabbed Dialog Box Class

Some handy dialog box tricks, tips and workarounds

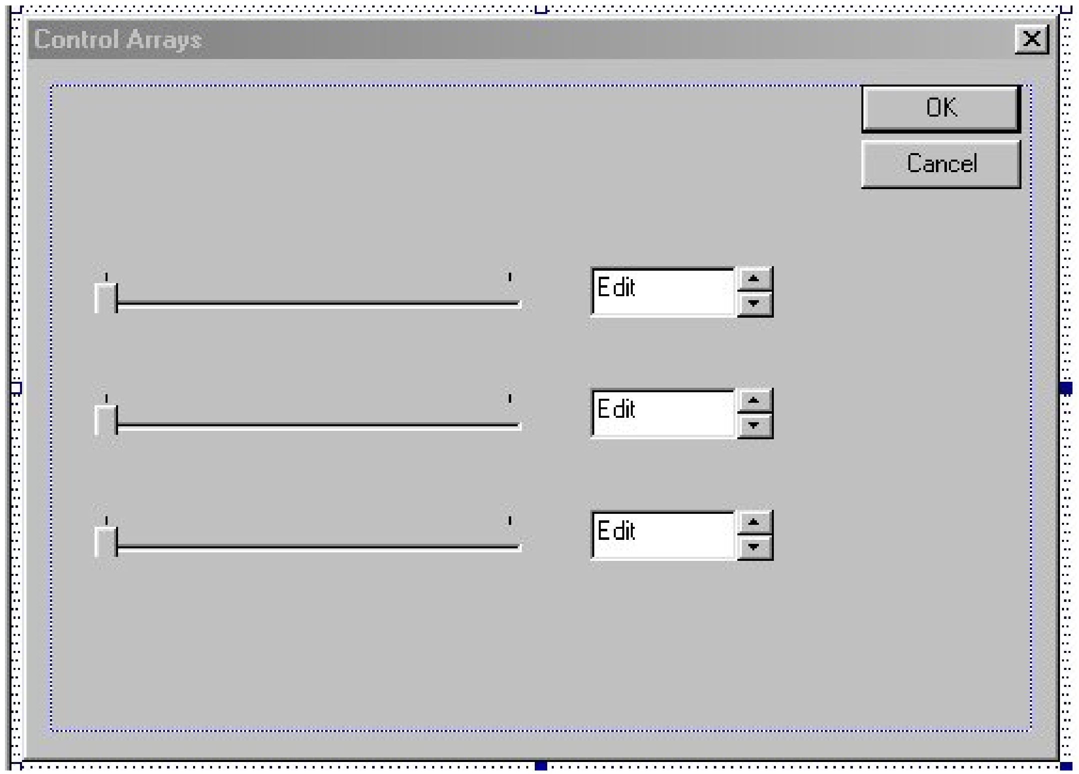
Dynamic Dialog Box

A class for child dialogs

WTL for MFC Programmers, Part IV - Dialogs and Controls

Dialog Boxes Guidelines

Adding controls to a dialog



With that done, now we advance onto the trickier part of the task.

Class Wizard (Next step)

This part can prove the most problematic. Because the wizard has either lost its coned hat or the wand has failed to deal with control arrays. Hence we have to do our own sorcery to work around it.

Now let's call up the Class Wizard leaving the Message Map. We want to select the following messages:

We want to map messages to all the spin controls and edit controls. The sliders we'll deal with later.

Make sure that Tab is on Message Maps. The class name is `CControlArrayDlg`. Select the message `WM_VSCROLL`. This automatically calls the function `OnVScroll`. This caters for all the vertical scrollers (spin button). Note **FormView apps also select the `WM_HSCROLL`. This automatically calls the function `OnHScroll` to handle the sliders**.

Tooltips in modal dialog boxes

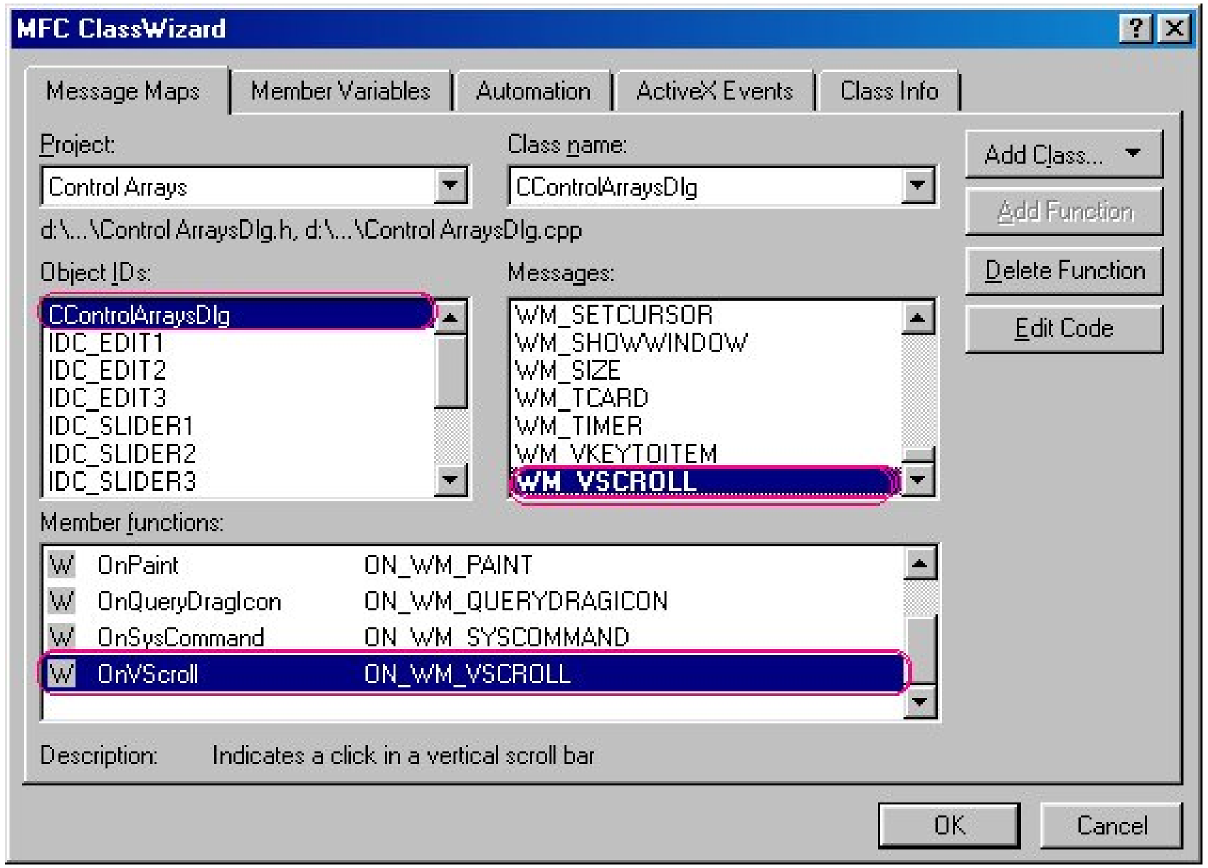
Controls Array

Use an ActiveX control in your Win32 Project without MFC with CreateWindowEx or in a dialog box

Custom Color Dialog Box

Dialog Boxes Management in WPF Application

Displaying a Dialog Box in SCSF


MFC ClassWizard

Project: Control Arrays

Class name: CControlArraysDlg

d:\...\Control ArraysDlg.h, d:\...\Control ArraysDlg.cpp

Object IDs:
- CControlArraysDlg
- IDC_EDIT1
- IDC_EDIT2
- IDC_EDIT3
- IDC_SLIDER1
- IDC_SLIDER2
- IDC_SLIDER3

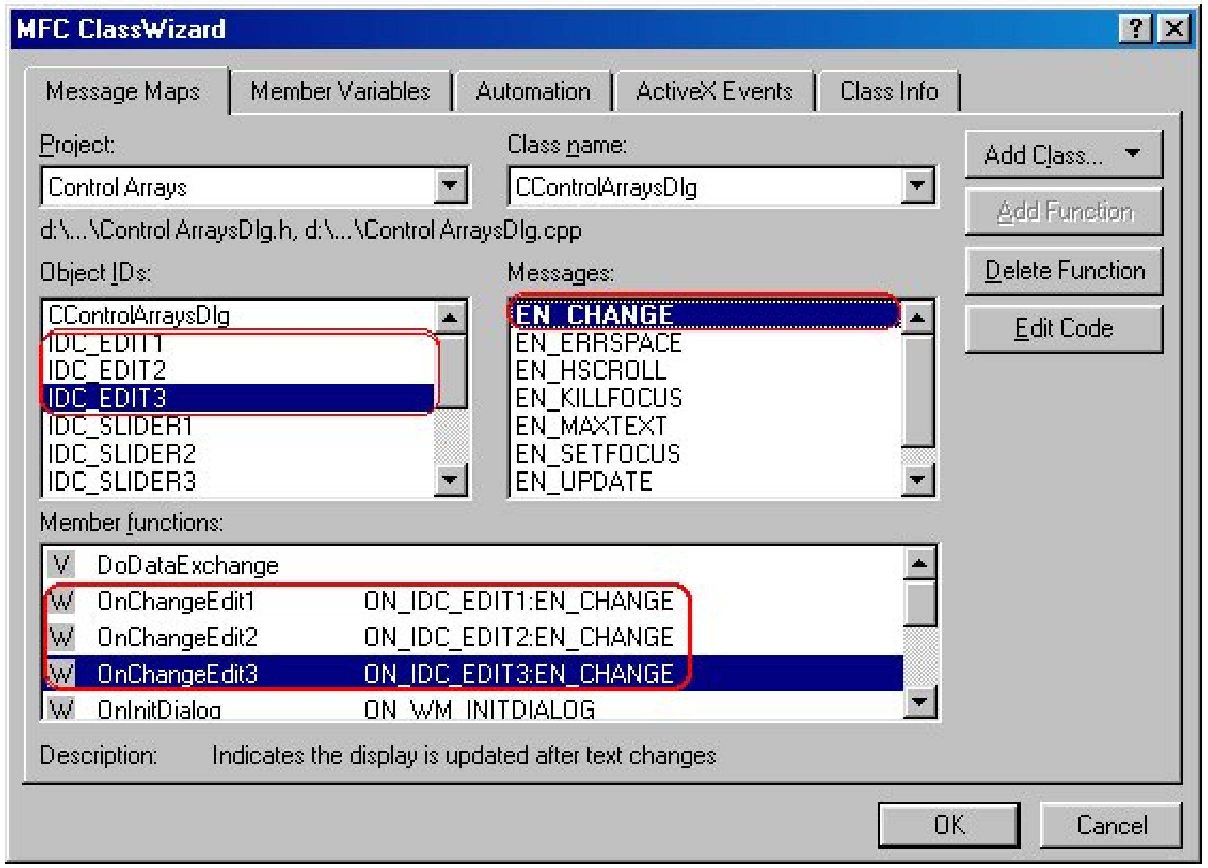
Messages:
- WM_SETCURSOR
- WM_SHOWWINDOW
- WM_SIZE
- WM_TCARD
- WM_TIMER
- WM_VKEYTOITEM
- WM_VSCROLL

Member functions:

W	OnPaint	ON_WM_PAINT
W	OnQueryDragIcon	ON_WM_QUERYDRAGICON
W	OnSysCommand	ON_WM_SYSCOMMAND
W	OnVScroll	ON_WM_VSCROLL

Description: Indicates a click in a vertical scroll bar

Select the control IDC_EDIT1. Select the message EN_UPDATE. Accept the suggested function name. Since EN messages only work on per edit control, repeat this for IDC_EDIT2 and IDC_EDIT3. Skip the edit change.



Project: Control Arrays

Class name: CControlArraysDlg

d:\...\Control ArraysDlg.h, d:\...\Control ArraysDlg.cpp

Object IDs:

```
CControlArraysDlg
IDC_EDIT1
IDC_EDIT2
IDC_EDIT3
IDC_SLIDER1
IDC_SLIDER2
IDC_SLIDER3
```

Messages:

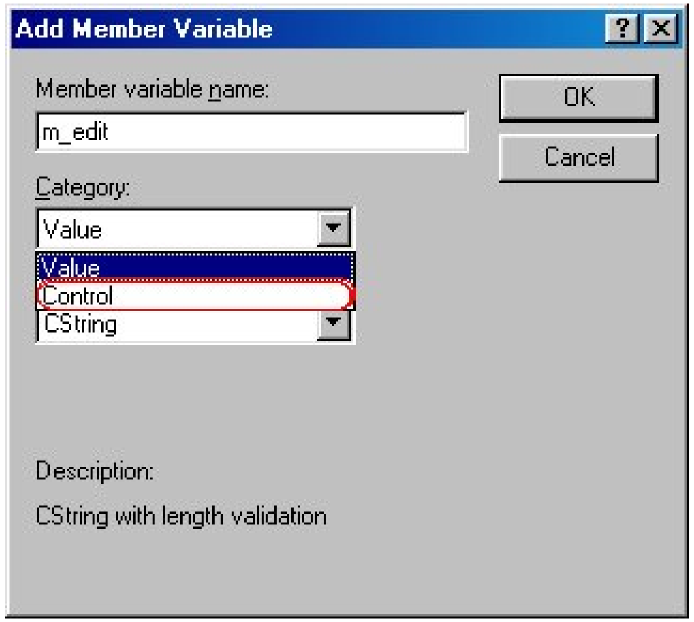
```
EN_CHANGE
EN_ERRSPACE
EN_HSCROLL
EN_KILLFOCUS
EN_MAXTEXT
EN_SETFOCUS
EN_UPDATE
```

Member functions:

```
V  DoDataExchange
W  OnChangeEdit1     ON_IDC_EDIT1:EN_CHANGE
W  OnChangeEdit2     ON_IDC_EDIT2:EN_CHANGE
W  OnChangeEdit3     ON_IDC_EDIT3:EN_CHANGE
W  OnInitDialog      ON_WM_INITDIALOG
```

Description: Indicates the display is updated after text changes

Select the Member Variable tab. Map variables to `IDC_EDIT1`, `IDC_SLIDER1` & `IDC_SPIN1` **only** (see note 1). The others will be dealt with in the next section. Set the slider and edit control of the category **control**. This is the default for the spin control.



Member variable name: m_edit

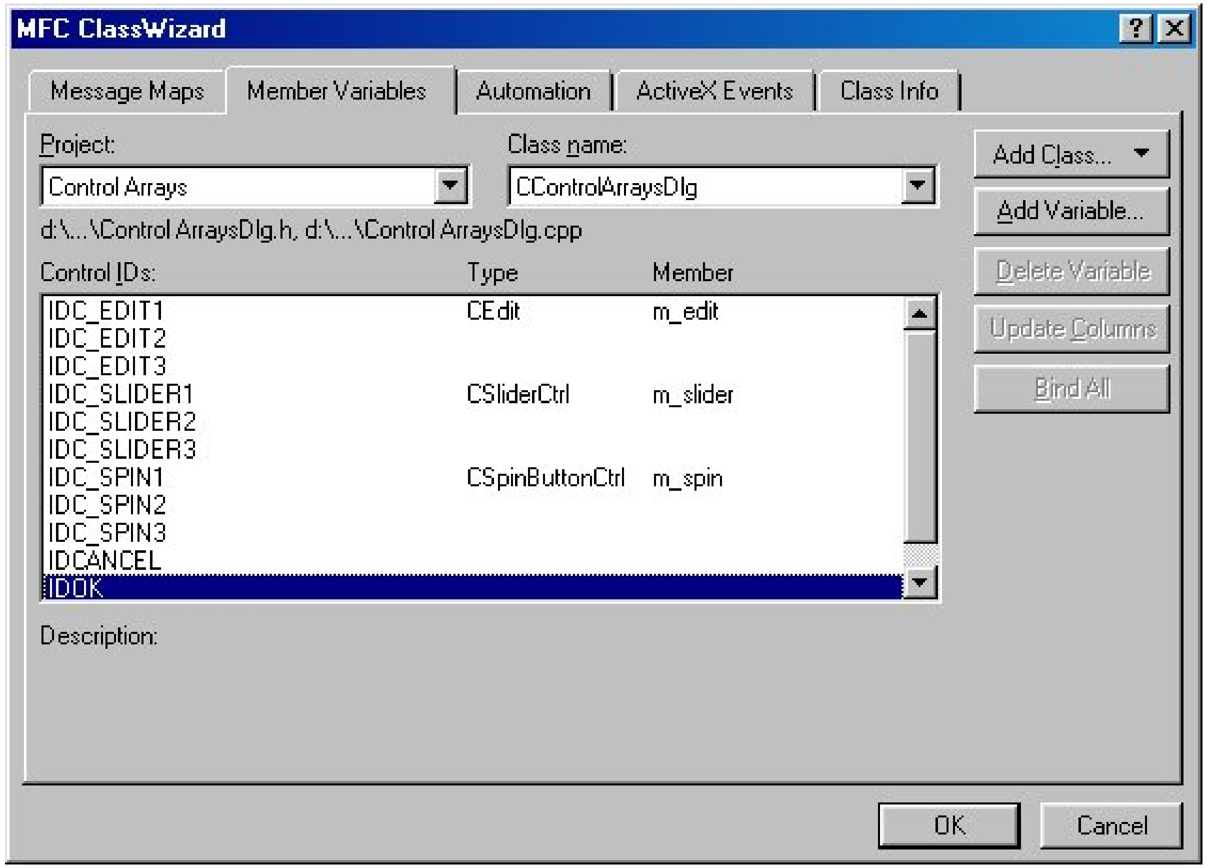
Category: Value

Variable type: CString

Description:

CString with length validation

Variable List



Project: Control Arrays

Class name: CControlArraysDlg

d:\...\Control ArraysDlg.h, d:\...\Control ArraysDlg.cpp

Control IDs:	Type	Member
IDC_EDIT1	CEdit	m_edit
IDC_EDIT2		
IDC_EDIT3		
IDC_SLIDER1	CSliderCtrl	m_slider
IDC_SLIDER2		
IDC_SLIDER3		
IDC_SPIN1	CSpinButtonCtrl	m_spin
IDC_SPIN2		
IDC_SPIN3		
IDCANCEL		
IDOK		

Description:

Now we're done with the Class Wizard. Click OK.

Coding the project

Sub sections

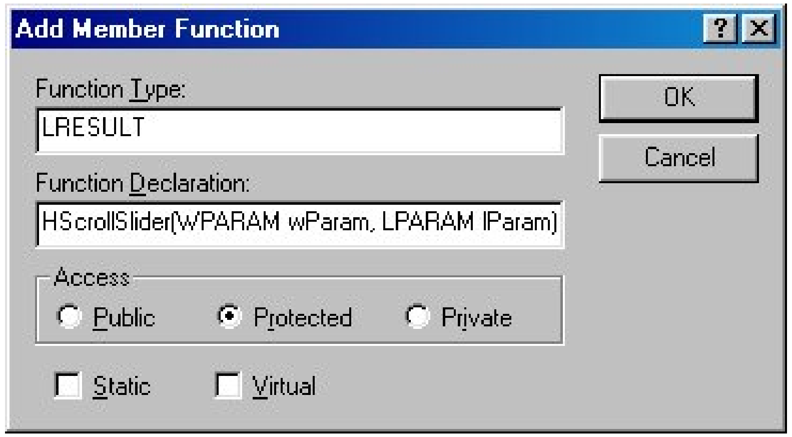
1. Message Mapping the dialog based sliders
2. Declaring the control arrays, connecting the variables to the controls
3. Coding the methods

Now onto the sexy part.

Message Mapping the sliders (For Dialog based apps ONLY)

Now firstly we'll map the sliders (see note 2) which requires quite a bit of coding. Select the Class View tab in the `ControlArrayDlg` and select the "Add member function" on the popup menu. Right click the function return of type `LRESULT` and declare it as `OnHScrollSlider(WPARAM wParam, LPARAM lParam)`. Finally, make its accessibility `protected`.

Visual description of how to add a function:



Add Member Function

Function Type: LRESULT

Function Declaration: HScrollSlider(WPARAM wParam, LPARAM lParam)

Access: Public [✓ Protected] Private

☐ Static ☐ Virtual

ControlArrayDemo1Dlg.cpp to find the text below:

Collapse | Copy Code

```
BEGIN_MESSAGE_MAP(CControlArraysDemo1Dlg, CDialog)
//{{AFX_MSG_MAP(CControlArraysDemo1Dlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_VSCROLL()
    ON_EN_CHANGE(IDC_EDIT1, OnChangeEdit1)
    ON_EN_CHANGE(IDC_EDIT2, OnChangeEdit2)
    ON_EN_CHANGE(IDC_EDIT3, OnChangeEdit3)
//}}AFX_MSG_MAP

END_MESSAGE_MAP()
```

Between `//}}AFX_MSG_MAP` and `END_MESSAGE_MAP()` insert `ON_MESSAGE(WM_HSCROLL, OnHScrollSlider)` to look like this:

⊟ Collapse | Copy Code

```
BEGIN_MESSAGE_MAP(CControlArraysDemo1Dlg, CDialog)
//{{AFX_MSG_MAP(CControlArraysDemo1Dlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_VSCROLL()
    ON_EN_CHANGE(IDC_EDIT1, OnChangeEdit1)
    ON_EN_CHANGE(IDC_EDIT2, OnChangeEdit2)
    ON_EN_CHANGE(IDC_EDIT3, OnChangeEdit3)
//}}AFX_MSG_MAP

// This is declared outside the AFX_MSG_MAP scope
// if later you decided to add more messags if will be deleted
ON_MESSAGE(WM_HSCROLL, OnHScrollSlider)

END_MESSAGE_MAP()
```

⊟ Collapse | Copy Code

```
afx_msg LRESULT OnHScrollSlider(WPARAM wParam, LPARAM lParam);
```

Now go to the file *ControlArraysDemo1Dlg.h* and find the text below:

⊟ Collapse | Copy Code

```
protected:
LRESULT OnHScrollSlider(WPARAM wParam, LPARAM lParam);
HICON m_hIcon;
```

Either drag and drop or cut and paste `LRESULT OnHScrollSlider(WPARAM wParam, LPARAM lParam);` between `//}}AFX_MSG` and `DECLARE_MESSAGE_MAP()` to look something like this:

⊟ Collapse | Copy Code

```
protected:
HICON m_hIcon;

// Generated message map functions
//{{AFX_MSG(CControlArraysDemo1Dlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnChangeEdit1();
    afx_msg void OnChangeEdit2();
    afx_msg void OnChangeEdit3();
//}}AFX_MSG

// This is declared outside the AFX_MSG scope
// if later you decided to add more messags if will be deleted
LRESULT OnHScrollSlider(WPARAM wParam, LPARAM lParam);

DECLARE_MESSAGE_MAP()
```

Staying in the header file we're going to work on some of the properties.

Declaring the control arrays, connecting the variables to the controls

In *ControlArraysDemo1Dlg.h* or *ControlArraysDemo2View.h* locate the snippet of code below:

⊟ Collapse | Copy Code

```
// Dialog Data
//{{AFX_DATA(CControlArraysDemo1Dlg)
    enum { IDD = IDD_CONTROLARRAYS_DIALOG };
    CSpinButtonCtrl    m_spin;
    CSliderCtrl    m_slider;
    CEdit    m_edit;
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CControlArraysDemo1Dlg)
```

Declare `m_spin` as an array of type `CSpinButtonCtrl`, `m_slider` as an array of type `CSliderCtrl` and `m_edit` as an array of type `CEdit` each having 3 elements, so that it looks like this:

⊟ Collapse | Copy Code

```
// Dialog Data
//{{AFX_DATA(CControlArraysDemo1Dlg)
    enum { IDD = IDD_CONTROLARRAYS_DIALOG };
    CSpinButtonCtrl    m_spin[3];
    CSliderCtrl    m_slider[3];
    CEdit    m_edit[3];
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CControlArraysDemo1Dlg)
    protected:
```

```
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL
```

That finished, we can get down to the "serious" coding.

Last part, coding the methods

Table of how the variables are connected to the controls:

Control ID	***Connecting variable***
`IDC_EDIT1`	`m_edit[0]`
`IDC_EDIT2`	`m_edit[1]`
`IDC_EDIT3`	`m_edit[2]`
`IDC_SLIDER1`	`m_slider[0]`
`IDC_SLIDER2`	`m_slider[1]`
`3IDC_SLIDER1`	`m_slider[2]`
`IDC_SPIN1`	`m_spin[0]`
`IDC_SPIN2`	`m_spin[1]`
`IDC_SPIN3`	`m_spin[2]`

Let's go to the file *ControlArrayDlg.cpp* and go to the function `DoDataExchange` using the table above. Modify the code below:

⊟ Collapse | Copy Code

```
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CControlArraysDemo1Dlg)
    DDX_Control(pDX, IDC_SPIN1, m_spin);
    DDX_Control(pDX, IDC_SLIDER1, m_slider);
    DDX_Control(pDX, IDC_EDIT1, m_edit);
//}}AFX_DATA_MAP
```

to

⊟ Collapse | Copy Code

```
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CControlArraysDemo1Dlg)
```

```
    DDX_Control(pDX, IDC_EDIT1, m_edit[0]);
    DDX_Control(pDX, IDC_EDIT2, m_edit[1]);
    DDX_Control(pDX, IDC_EDIT3, m_edit[2]);
    DDX_Control(pDX, IDC_SLIDER1, m_slider[0]);
    DDX_Control(pDX, IDC_SLIDER2, m_slider[1]);
    DDX_Control(pDX, IDC_SLIDER3, m_slider[2]);
    DDX_Control(pDX, IDC_SPIN1, m_spin[0]);
    DDX_Control(pDX, IDC_SPIN2, m_spin[1]);
    DDX_Control(pDX, IDC_SPIN3, m_spin[2]);
//}}AFX_DATA_MAP
```

The coding now starts:

Create a protected function declared as: `BOOL StringToNumber(LPCTSTR szText, int &nValue)` to convert the `CString` to a number below the code:

⊟ Collapse | Copy Code

```
BOOL CControlArraysDemo1Dlg::StringToNumber(LPCTSTR pszText, int &nValue)
{
    if(!pszText)
        return FALSE;

    while(*pszText)
    {
        nValue *= 10;
        if((*pszText < 48)||(  *pszText >  57))
        {
            nValue = 0;
            return FALSE;
        }

        nValue += *pszText - 48;
        pszText++;
    }

    return TRUE;

}
```

Code for handling the spin controls:

⊟ Collapse | Copy Code

```
void CControlArraysDemo1Dlg::OnVScroll(UINT nSBCode,
                          UINT nPos, CScrollBar* pScrollBar)
```

```
{
    // nSBCode is NOT needed because the scroller in the class is spin
    // First convert the pointer to one of type CSpinButtonCtrl
    CSpinButtonCtrl *pSpin = reinterpret_cast<CSPINBUTTONCTRL *>(pScrollBar);
    int nIndex = pSpin - &m_spin[0];    // Get the actual index of spin control
    CString szValue;            // The string for the edit box
    szValue.Format("%d",nPos);    // Format it that number is a string

    m_bChangedByCtrl = TRUE;    //Edit box is updated a control Flag it TRUE
    m_edit[nIndex].SetWindowText(szValue);    //Self explanatory
    m_slider[nIndex].SetPos(nPos);    //ditto

    CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
}
```

The work around for handling the slider control (**Dialog based apps only**):

⊟ Collapse | Copy Code

```
LRESULT CControlArraysDemo1Dlg::OnHScrollSlider(WPARAM wParam, LPARAM lParam)
{
    // is wParam NOT needed because message is called by a control
    HWND hwnd;        // Interrogator of the slider's handle
    hwnd = (HWND)lParam;    //Casting to a window hadnd
    CString szValue;        // The string for the edit box
    int nIndex;        // For obtaining index of slider control
    int nPos;        //to obtain the slider's position

    //Is it m_slider[0]?
    if(m_slider[0].m_hWnd == hwnd)
    {
        nIndex = 0;    // Yes! then index is 0
    }

    // No, see if it's m_slider[1]?
    else if(m_slider[1].m_hWnd == hwnd)
    {
        nIndex = 1;    // Yes! then index is 1
    }

    else
    {
        nIndex = 2;    // Obviously, the index is 2
    }

    nPos = m_slider[nIndex].GetPos();    // Self explanatory
    szValue.Format("%d",nPos);    // Format it that number is a string
    m_edit[nIndex].SetWindowText(szValue);    // Again self explanatory
    m_spin[nIndex].SetPos(nPos);    //ditto
    return 0;

}
```

Alternatively in the FormView (CControlArrayDemo2View):

⊟ Collapse | Copy Code

```
void CControlArraysDemo2View::OnHScroll(UINT nSBCode,
                        UINT nPos, CScrollBar* pScrollBar)
{
    // Let's ascertain whether or not the slider was slidden
    if(pScrollBar)
    {
        CSliderCtrl *pSlider = reinterpret_cast<CSLIDERCTRL *>(pScrollBar);
        int nIndex = pSlider - &m_slider[0];
        int nValue = m_slider[nIndex].GetPos();
        CString szValue;
        szValue.Format("%d",nValue);
        m_spin[nIndex].SetPos(nValue);
        m_edit[nIndex].SetWindowText(szValue);
    }

    else
    {
        // Your code for handling the conventional scroll bar
    }

    CFormView::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

Handling the updates to IDC_EDIT1:

⊟ Collapse | Copy Code

```
void CControlArraysDemo1Dlg::OnUpdateEdit1()
{
    // Check to see if the control is focused
    // if so modify the other controls at index 0
    // if not no further action is taken
    if(GetFocus() == &m_edit[0])
    {
        CString szText;
        int nValue = 0;
        m_edit[0].GetWindowText(szText);
        StringToNumber(szText,nValue);

        m_slider[0].SetPos(nValue);
        m_spin[0].SetPos(nValue);
    }

}
```

Code for `OnUpdateEdit2 (IDC_EDIT2)` and `OnUpdateEdit3 (IDC_EDIT3)` are very similar. The differences are in the indexes of the other controls.

Finally code that does the intialising which is called from `OnInitDialog`:

Collapse | Copy Code

```
void CControlArraysDemo1Dlg::Initialisation()
{
    int nCounter;

    for(nCounter = 0;  nCounter <  3; nCounter++)
    {
        m_edit[nCounter].ModifyStyle(0,ES_NUMBER);
        m_edit[nCounter].SetWindowText(_T("0"));
        m_edit[nCounter].SetLimitText(2);
        m_slider[nCounter].SetRange(0,99);
        m_slider[nCounter].SetTicFreq(10);
        m_slider[nCounter].SetPos(0);
        m_spin[nCounter].SetRange(0,99);
        m_spin[nCounter].SetPos(0);
    }

}
```

Hey presto! There you have it, a working dialog based app that uses control arrays on the dialogue.

Points of note

Here are the problems that you will encounter when using control arrays:

1. When Class Wizard is called it issues the message: *Parsing error, ";". Input line CSpinButtonCtrl m_spin[3]* (in this case). A possible workaround is to substitute [for a double-underscore and] for

the gobbledegook characters using Replace all, commenting out some of the code.

2. I don't know the reason behind it, is it the call to parent class the culprit? The message `WM_HSCROLL` with slides tend to reset the control. Although, it's being slid to a different position.

Conclusion

I don't know if you agree with me. I find arrays can make coding a heck of a lot easier. Although this demo only uses three controls, this can be extended to other controls. You may see the reasons I stated in the introduction.

History

- Version 1. (Feb 2005) Original app.
- Version 2. (August 2005) Bug fix, problems running the code on a FormView dealt with. `EN_UPDATE` is used as opposed to `EN_CHANGE`.

License

This article has no explicit license attached to it but may contain usage terms in the article text or the download files themselves. If in doubt please contact the author via the discussion board below.

A list of licenses authors might use can be found here

About the Author

Alton Williams

United Kingdom
Member

I've currently studying Computer Science at uni in London.
After coding in BASIC in the 80s with a bit of COBOL (yuk to the latter)

Since starting my degree I'm having a love affair with a language called C++.

I'm also learning MFC

Article Top Like 0 0 Tweet 0

Rate this: *Poor* ◯ ◯ ◯ ◯ ◯ *Excellent* Vote

Comments and Discussions

Add a Comment or Question ? Search this forum Go

☐ Profile popups Spacing Relaxed Noise Medium Layout Normal Per page 25 Update

First Prev Next

Could you elaborate note1, please	**CiaChing**	**15:16 18 Jan '07**
Re: Could you elaborate note1, please [modified]	**Alton Williams**	3:18 18 May '07
CFormView HScroll bar	**clive04**	**23:42 23 Aug '05**
Re: CFormView HScroll bar	**Alton Williams**	22:45 25 Aug '05

Suggestions	**KarstenK**	**18:18 2 Mar '05**
Re: Suggestions	rbid	20:04 2 Mar '05
Re: Suggestions	Alton Williams	21:33 2 Mar '05
Re: Suggestions	Plennguyen	15:30 12 Aug '06
Re: Suggestions	rbid	23:55 14 Aug '06
Re: Suggestions	Alton Williams	22:09 2 Mar '05
Re: Suggestions (Subclassing!!0	Alton Williams	3:29 18 May '07

Last Visit: 20:18 9 Jan '13   Last Update: 16:17 9 Jan '13   Refresh   1

General   News   Suggestion   Question   Bug   Answer   Joke   Rant   Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.