# Automation part 2

Program examples compiled using Visual C++ 6.0 compiler on Windows XP Pro machine with Service Pack 2. The Excel version is Excel 2003/Office 11. Topics and sub topics for this tutorial are listed below. Don't forget to read Tenouk's small <u>disclaimer</u>. The supplementary notes for this tutorial are <u>mymfc29C.xls</u>, <u>automation</u>, <u>variant</u> and <u>COlevariant class</u>.

## Index:

The MYMFC29C SDI Automation Component - EXE Example: with User Interface MYMFC29C From Scratch CMymfc29CDoc class CMainFrame class CMymfc29CView class The Story

# The MYMFC29C SDI Automation Component EXE Example: with User Interface

This Automation component example illustrates the use of a document component class in an **SDI application** in which a new process is started for each object. This component program demonstrates an indexed property plus a method that constructs a new COM object.

The first Automation component example you saw, MYMFC29A, didn't have a user interface. The global class factory constructed a CBank object that did the component's work. What if you want your EXE component to have a window? If you've bought into the MFC document-view architecture, you'll want the document, view, and frame, with all the benefits they provide.

Suppose you created a regular MFC application and then added a COM-creatable class such as CBank. How do you attach the CBank object to the document and view? From a CBank class member function, you could navigate through the application object and main frame to the current document or view, but you'd have a tough time in an MDI application if you encountered several component objects and several documents. There is a better way. You make the document class the **creatable class**, and you have the full support of AppWizard for this task. This is true for both MDI and SDI applications.

The MDI MSDN's <u>Autoclik</u> example demonstrates how COM triggers the construction of new document, view, and child frame objects each time an Automation client creates a new component object. Because the MYMFC29C example is an SDI program, Windows starts a new process each time the client creates an object. Immediately after the program starts, COM, with the help of the MFC application framework, constructs not only the Automation-aware document but also the view and the main frame window.

Now is a good time to experiment with the MYMFC29C application, which was first generated by AppWizard with the **Automation** option checked. It's a Windows-based alarm clock program designed to be manipulated from an Automation client such as Excel. MYMFC29C has the following **properties** and **methods**.

| Method/property | Description   |  |
|-----------------|---|--|
| Time            | DATE property that holds a COM DATE (m_Time)                                    |  |
| Figure          | Indexed VARIANT property for the four figures on the clock face                 |  |
| Figure          | (m_strFigure[])   |  |
| Defeceshui      | Method that invalidates the view window and brings the main frame window to the |  |
| Rellesiiwiii    | top(Refresh)  |  |
| ShowWin         | Method that displays the application's main window (ShowWin)                    |  |
| CreateAlarm     | Method that creates a CAlarm object and returns its IDispatch pointer           |  |
|                 | (CreateAlarm)   |  |

Table 1.

Here are the steps for building and running an already completed MYMFC29C.

From Visual C++, open the workspace mymfc29C.dsw or any directory that you have designated for your project. Build the project to produce the **mymfc29C.exe** file in the project's Debug subdirectory.

Run the program once to register it. The program is designed to be executed either as a **standalone application** or as an **Automation component**. When you run it from Windows or from Visual C++, it updates the Registry automatically and displays the face of a clock with the characters XII, III, VI, and IX at the 12, 3, 6, and 9 o'clock positions. Exit the program.

Next, load the mymfc29C.xls Excel workbook file. The worksheet should look something like the one shown here.

|      | licro | soft Ex      | cel - | mymfc2 | 9C.xls          |               |              |                |      |                  |         |       |           |
|------|-------|--------------|-------|--------|-----------------|---------------|--------------|----------------|------|------------------|---------|-------|-----------|
| :2   | Eile  | <u>E</u> dit | ⊻iew  | Insert | F <u>o</u> rmat | <u>T</u> ools | <u>D</u> ata | <u>W</u> indow | Hel  | p Ado <u>b</u> e | PDF     | _ é   | 7 X       |
| ! D  | 2     | 2            | 9     | a 📖    | - 🖺 -           | 19 -          | 😣 Σ          | - AZ↓          |      | 0 🔋 i            | Arial   |       | - 11<br>- |
| : 🔁  |       | 78 📮         | Aa    | ab [ 🗳 |                 | •             |              | 4 4            | 1    | ۲                | J   🔢 1 | Ţ     |           |
|      | E3    |              | •     | fx -   | 10:00:00        | AM            |              |                |      |                  |         |       |           |
|      |       | А            |       | В      | C               |               | D            | E              |      | F                | G       | H     |           |
| 1    | l     | Load Cl      | lock  |        | Refresh         | Time          |              | Set            | Alar | m                | Unload  | Clock | HE        |
| 3    | XII   |              |       |        | VI              | DX            | (            | 10:00 /        | AM   |                  |         |       |           |
| 4    |       |              |       |        |                 |               |              |                |      |                  |         |       | - 1       |
| 5    |       |              |       |        |                 |               |              |                |      |                  |         |       | - 1       |
| 6    |       |              |       |        |                 |               |              |                |      |                  |         |       | ~         |
| H A  | •     | • ∖ She      | et1,  | Sheet2 | / Sheet3        | 1             |              | <              |      |                  |         | >     |           |
| Read | ły    |              |       |        |                 |               |              | _              |      |                  |         |       |           |

Figure 1: mymfc29C.xls, a client, to test MYMFC29C component.

Click the **Load Clock** button, and then double-click the **Set Alarm** button. There could be a long delay after you click the **Load Clock** button, depending on your system. The clock should appear as shown below, with the letter **A** indicating the alarm setting.



Figure 2: Alarm is set through a client.

If you've started the component program from the debugger, you can watch the Debug window to see when InitInstance() is called and when the document object is constructed.

# MYMFC29C From Scratch

To experience a real programming, the following are steps to build MYMFC29C from scratch. Have a good practice! This is an SDI application with Automation support.

| New  | ? 🛛   |
|--|---|
| Files       Projects       Workspaces       Other Documents         ATL COM AppWizard       Win32 Static Library         Cluster Resource Type Wizard       Win32 Static Library         Custom AppWizard       Database Project         Database Project       DevStudio Add-in Wizard         Extended Stored Proc Wizard       ISAPI Extension Wizard         Makefile       MFC ActiveX ControlWizard         MFC AppWizard (dll)       MFC AppWizard (exe)         New Database Wizard       Win32 Application         Win32 Console Application       Win32 Dynamic-Link Library | Project name:<br>mymfc29Q<br>Logation:<br>F:\MFCPROJECT\mymfc29C<br><br>Create new workspace<br>Add to current workspace<br>Dependency of:<br>Platforms:<br>Win32 |
|  | OK Cancel   |

Figure 3: Visual C++ new project dialog.

Select the **Single Document**.

| MFC AppWizard - Step 1  | ? 🛛   |
|---|---|
| Application - Document 1         File Edit View Window Help         Beady | <ul> <li>What type of application would you like to create?</li> <li>Single documenti</li> <li>Multiple documents</li> <li>Dialog based</li> <li>✓ Document/View architecture support?</li> <li>What language would you like your resources in?</li> <li>English [United States] (APPwZENU.DLL ▼</li> </ul> |
| < <u>B</u> ack  | <u>N</u> ext > <u>F</u> inish Cancel  |

Figure 4: MYMFC29C – AppWizard step 1 of 6, SDI application.

| MFC AppWizard - Step 2 of 6  | ? 🔀  |
|--|--|
| Application         File Edit Yiew Window Help         Mew         Open         Close         Sare         Sare         Sare         Peady | <ul> <li>What database support would you like to include?</li> <li>None</li> <li>Header files only</li> <li>Database view without file support</li> <li>Database view with file support</li> <li>If you include a database view, you must select a data source.</li> <li>Data Source</li> <li>No data source is selected.</li> </ul> |
| < <u>B</u> ack   | <u>N</u> ext > <u>F</u> inish Cancel   |

Figure 5: MYMFC29C – AppWizard step 2 of 6.

Don't forget to select **Automation** option.

| MFC AppWizard - Step 3 of 6 | ? 🛛  |  |  |  |
|-----------------------------|--|--|--|--|
| Application                 | What compound document support would you like to<br>include? |  |  |  |
|                             | None   |  |  |  |
|                             | C <u>C</u> ontainer  |  |  |  |
|                             | Mini-server  |  |  |  |
|                             | ◯ Full- <u>s</u> erver                                       |  |  |  |
|                             | O Both container and server                                  |  |  |  |
| Ready                       | Active <u>d</u> ocument server                               |  |  |  |
|                             | Active document container                                    |  |  |  |
|                             | Would you like support for compound files?                   |  |  |  |
|                             | <ul> <li>Yes, please</li> <li>Ma, therefore way</li> </ul>   |  |  |  |
|                             | 💿 No, thank you  |  |  |  |
|                             | What other support would you like to include?                |  |  |  |
|                             | <ul> <li>✓ Automation</li> <li>✓ ActiveX Controls</li> </ul> |  |  |  |
|                             |  |  |  |  |
| < <u>B</u> ack              | <u>N</u> ext > <u>F</u> inish Cancel                         |  |  |  |

Figure 6: MYMFC29C – AppWizard step 3 of 6.

| MFC AppWizard - Step 4 of 6   | ? 🗙   |
|---|---|
| Application   | What features would you like to include?  |
| File Edit Yiew Window Help         Print Pregiew         Print Pregiew         Print Setup         Exit         Ready         Editing Constrol:         Record         X       Check Boz         Radio Button | <ul> <li>Docking toolbar</li> <li>Initial status bar</li> <li>Printing and print preview</li> <li>Context-sensitive Help</li> <li>3D controls</li> <li>MAPI (Messaging API)</li> <li>Windows Sockets</li> <li>How do you want your toolbars to look?</li> <li>Normal</li> <li>Internet Explorer ReBars</li> </ul> |
| Back  | How many files would you like on your recent file list?  4   Advanced  Next > Finish Cancel   |

Figure 7: MYMFC29C – AppWizard step 4 of 6.

Click the **Advanced** button. Here, we just use the default settings for **Localized strings**. Take note that the **Doc type name** has the maximum six characters.

| Advanced Options                                   | ? 🛛  |
|--|--|
| Document Template Strings<br>Non-localized strings | Window Styles  |
| File e <u>x</u> tension:                           | File type <u>I</u> D:<br>Mymfc29C.Document                 |
| Localized strings                                  |  |
| Language:<br>English [United States]               | Main frame <u>c</u> aption:<br>mymfc29C                    |
| Doc type name:<br>mfc29C                           | <u>F</u> ilter name:                                       |
| File <u>n</u> ew name (short<br>name):<br>mfc29C   | File type na <u>m</u> e (long<br>name):<br>mfc29C Document |
|  | Close  |

| Figura S | 2. MVME       | $C^{20C}$    | AnnWiza   | rd default | L ocalizad | etringe |
|----------|---------------|--------------|-----------|------------|------------|---------|
| riguit ( | 5. IVI I IVII | $C_{2}C = I$ | npp willa | lu uciaun  | Localizeu  | su mgs. |

| MFC AppWizard - Step 5 of 6   |  |  | ? 🔀  |
|---|--|--|--|
| • Microsoft Developer Studio         File Edit Yiew Issert Bwild Help         • Project         • Project         • • • • • • • • • • • • • • • • • • • | What style of<br>MFC St<br>Windov<br>Would you like<br>Yes, pla<br>No, tha<br>How would you<br>As a sh<br>As a sta | project would you<br>andard<br>vs Explorer<br>e to generate sour<br>ease<br>nk you<br>ou like to use the M<br>ared <u>D</u> LL<br>atically linked librar | like ?<br>ce file comments?<br>IFC library?<br>Y |
| < <u>B</u> ack  | <u>N</u> ext >   | <u>F</u> inish   | Cancel   |

Figure 9: MYMFC29C – AppWizard step 5 of 6.

| MFC AppWizard - Step 6 of 6 |  | ? 🔀                               |
|-----------------------------|--|-----------------------------------|
|                             | AppWizard creates the I<br>CMymfc29CView<br>CMymfc29CApp<br>CMainFrame<br>CMymfc29CDoc | ollowing <u>c</u> lasses for you: |
|                             | Class name:  | H <u>e</u> ader file:             |
|                             | CMymfc29CView  | mymfc29CView.h                    |
|                             | B <u>a</u> se class:   | Implementation file:              |
|                             | CView 💌  | mymfc29CView.cpp                  |
|                             |  |                                   |
| < <u>B</u> ack              | Next > <u>F</u> in   | ish Cancel                        |

Figure 10: MYMFC29C – AppWizard step 6 of 6.

| New Project Information   |
|---|
| AppWizard will create a new skeleton project with the following specifications:   |
| Application type of mymfc29C:<br>Single Document Interface Application targeting:<br>Win32  |
| Classes to be created:<br>Application: CMymfc29CApp in mymfc29C.h and mymfc29C.cpp<br>Frame: CMainFrame in MainFrm.h and MainFrm.cpp<br>Document: CMymfc29CDoc in mymfc29CDoc.h and mymfc29CDoc.cpp<br>View: CMymfc29CView in mymfc29CView.h and mymfc29CView.cpp |
| Features:<br>+ 3D Controls<br>+ Uses shared DLL implementation (MFC42.DLL)<br>+ Automation support enabled<br>+ ActiveX Controls support enabled<br>+ Localizable text in:<br>English [United States]   |
| Project Directory:<br>F:\MFCPR0JECT\mymfc29C  |
| (OK) Cancel   |

Figure 11: MYMFC29C project summary.

Let create our own icon for this project. Before that, disable the IDR\_MFC29CTYPE icon by enclosing it in double quotes.

| Icon Properties      | X |
|----------------------|---|
| HA 😵 Resource        |   |
| ID: "IDR_MFC29CTYPE" |   |

Figure 12: Disabling the default icon.

You can verify it through the **View Resource Symbols** menu.

| Eile Edit     | <u>View</u> Insert Project Build Too |
|---------------|--------------------------------------|
| 🗎 🚅 E         | 💦 Class <u>W</u> izard Ctrl+W        |
| CMyatltestDlg | ID= Resource Symbols                 |
|               | Resource I <u>n</u> cludes           |
| 🖃 🚭 mya       | 🖹 Full Screen                        |
|               | Wor <u>k</u> space Alt+0             |
| ÷             | <u>O</u> utput Alt+2                 |
| ÷ 🚞 ۱         | Debug Windows                        |
|               | 🕼 Refres <u>h</u>                    |
|               | Properties Alt+Enter                 |

Figure 13: Viewing resource symbols.

| Resource Symbols  |                                   |        | ? 🔀  |
|---|-----------------------------------|--------|--|
| N <u>a</u> me<br>IDD_ABOUTBOX<br>IDP_OLE_INIT_FAILED<br>IDR_MAINFRAME<br>IDR_MFC29CTYPE | Value<br>100<br>100<br>128<br>129 | In Use | Close<br><u>N</u> ew<br>Delete<br>Change<br>⊻iew Use |
| <ul> <li>Show read-only symbols</li> <li>Used by:</li> <li>-not used-</li> </ul>        |                                   |        |  |

#### Figure 14: MYMFC29C resource symbols.

Modify the icon for both the Standard (32 x 32) and Small (16 x 16).



Figure 15: Standard icon.



Figure 16: Small icon.

Create a new CAlarm class using ClassWizard.

| MFC ClassWizard   | ? 🔀  |
|---|--|
| Message Maps       Member Variables       Automation       ActiveX Events       Class Info         Project:       Class name:         Immfc29C       CAboutDlg         F:\\mymfc29C\mymfc29C.cpp         Object [Ds:       Messages:         CAboutDlg       CalcWindowRect         ID_APP_ABOUT       CalcWindowProc         ID_APP_EXIT       DefWindowProc         ID_EDIT_COPY       DoD ataE schange         ID_EDIT_PASTE       OModal         ID_EDIT_UNDO       Member functions: | Add Class ▼<br>New<br>From a type library<br>Edit Code |
| Description:  | Cancel   |

Figure 17: Creating and adding new class to a project using ClassWizard.

Don't forget to select the Automation support option.

| New Class  |                                     | ? 🔀          |
|--|-------------------------------------|--------------|
| Class information<br><u>N</u> ame:<br>File name:   | CAlarm<br>Alarm.cpp<br>Change       | OK<br>Cancel |
| <u>B</u> ase class:                                | CCmdTarget 💌                        |              |
| <u>D</u> ialog ID:                                 | <b>v</b>                            |              |
| The base class                                     | does not require a dialog resource. |              |
| Automation<br>None<br>Automation<br>C Createable t | by type ID: mymfc29C.Alarm          |              |

Figure 18: Entering CAlarm class information.

Let do a clean-up by rebuilding the ClassWizard database. Delete the CLW file under the project directory.

| mymfc29C.clw              |                  | 2 KB CLV            |
|---------------------------|------------------|---------------------|
| 📑 mymfc29C.cp             | Open             |                     |
| 🚰 mymfc29C.ds             | Upload using W:  | 5_FTP Upload Wizard |
| 🥙 mymfc29C.ds             | E-mail with Yaho | 00!                 |
| h mymfc29C.h              | Send To          | •                   |
| 🖭 mymfc29C.nc 🗕           | Joing to         | •                   |
| 🖹 mymfc29C.oc             | Cut              |                     |
| 🖻 mymfc29C.op             | Сору             |                     |
| Emymfc29C.rc              | Create Shortcut  |                     |
| mymfc29C.re               | Delete           |                     |
| mymtc29CDo                | Rename           |                     |
| mymfc29CDo<br>mymfc29CVie | Properties       |                     |

Figure 19: Deleting the ClassWizard database file.

Then launch ClassWizard, the following prompt will be displayed. Just click the **Yes** button for the following prompt.



Figure 20: Building ClassWizard database prompt.

And click the **OK** button for the following dialog.

| Select Source Files - mym                        | ifc29C.clw  |          | ? 🔀             |
|--|---|----------|-----------------|
| File <u>n</u> ame:                               | Directories:<br>f:\mfcproject\mymfc29c<br>f:\<br>mfcproject<br>mymfc29C<br>Debug<br>res |          | OK<br>Cancel    |
| mymfc29CDoc.h                                    | Dri <u>v</u> es:  | ~        |                 |
| Source Files (*.h;*.cpp) 💌                       | f: hd0c   | •        |                 |
| <u>Files in project:</u>                         |   |          |                 |
| F:\MFCPROJECT\MYMFC290<br>F:\MFCPROJECT\MYMFC290 | CVAlarm.cpp<br>CVAlarm.h  | <u>^</u> | Add             |
| F:\MFCPR0JECT\MYMFC290                           | MainFrm.cpp   |          | Add All         |
| F:\MFCPROJECT\MYMFC290<br>F:\MFCPROJECT\MYMFC290 | Nymfc29C.cpp<br>Nymfc29C.h  | ~        | Re <u>m</u> ove |

Figure 21: ClassWizard database building files inclusion dialog.

| MFC ClassWizard   | ? 🛛               |
|---|-------------------|
| Message Maps   Member Variables   Automation   ActiveX Events   Class Info          | 1                 |
| Project: Class <u>n</u> ame:  | Add Class 🔻       |
| F:\mfcproject\mymfc29C\Alarm.h, F:\\mymfc29C\Alarm.cpp                              | Add Function      |
| Object IDs: Messages:   | Delete Function   |
| CAlarm ID_APP_ABOUT ID_APP_EXIT ID_EDIT_COPY ID_EDIT_CUT ID_EDIT_PASTE ID_EDIT_UNDO | <u>E</u> dit Code |
| Member <u>f</u> unctions:           V         OnFinalRelease                        |                   |
| Description:  |                   |
|   | K Cancel          |

Figure 22: CAlarm class in ClassWizard.

Add property to CAlarm without notification function. Select the **Automation** tab of the ClassWizard. Then click the **Add Property** button.

| MFC ClassWizard                                      |  |   |            | ? 🗙  |
|--|--|---|------------|--|
| Message Maps       Member Variables         Project: | Automation<br>Class <u>n</u> ame<br>CAlarm<br>ymfc29CVAlarm<br>Select a cl<br>and click A<br>add functio<br>Both Add N<br>you to add | ActiveX Events<br>e:<br>h.cpp,<br>ass that supports Au<br>Add Property or Add M<br>onality to your interfac<br>Method and Add Proj<br>stock and custom in | Class Info | Add Class  Add Method Add Property Delete Edit Code Data Binding |
| Default property                                     |  |   |            |  |
|  |  |   | OK         | Cancel   |

Figure 23: Adding property and method through Automation page of ClassWizard.

Add Time property, fill up as shown below.

| Add Property           |                         |                           | ? 🔀    |
|------------------------|-------------------------|---------------------------|--------|
| <u>E</u> xternal name: | Time                    | •                         | OK     |
| <u>T</u> ype:          | DATE                    | •                         | Cancel |
| Variable <u>n</u> ame: | m_time                  |                           |        |
| Notification function: | OnTimeChanged           |                           |        |
| _ Implementation       |                         |                           |        |
| C Stock @              | Member <u>v</u> ariable | ◯ Get/Set <u>m</u> ethods |        |
| Parameter list:        |                         |                           |        |
| Name                   | Туре                    |                           |        |
|                        |                         |                           |        |
|                        |                         |                           |        |
|                        |                         |                           |        |

Figure 24: Adding Time property.

In **Alarm.h**, replace:

```
DECLARE_DYNCREATE(CAlarm)
```

with

DECLARE\_DYNAMIC(CAlarm)

```
// CAlarm command target
class CAlarm : public CCmdTarget
{
    DECLARE_DYNAMIC(CAlarm)
    CAlarm(); // protected
```

Listing 1.

Modify the protected constructor to become public (or you can use friend keyword) and add the following parameter.

DATE time

```
class CAlarm : public CCmdTarget
{
    DECLARE_DYNAMIC(CAlarm)
public:
    CAlarm(DATE time); // public constructor
// Attributes
public:
```

Listing 2.

In Alarm.cpp change:

IMPLEMENT\_DYNCREATE(CAlarm, CCmdTarget)

to

IMPLEMENT\_DYNAMIC(CAlarm, CCmdTarget)

🛛 🖊 CAlarm

IMPLEMENT\_DYNAMIC(CAlarm, CCmdTarget)

Listing 3.

Then, modify the CAlarm constructor as shown below.

```
CAlarm::CAlarm(DATE time)
{
    TRACE("Entering CAlarm ctor -- vaTime = %f\n", (double) time);
    m_time = time;
    EnableAutomation();
}
```

// CAlarm

```
IMPLEMENT_DYNAMIC(CAlarm, CCmdTarget)
CAlarm::CAlarm(DATE time)
{
    TRACE("Entering CAlarm ctor -- vaTime = %f\n", (double) time);
    m_time = time;
    EnableAutomation();
}
```

Listing 4.

Add the following **properties** and **methods** for the CMymfc29CDoc class through the **Automation** tab of the ClassWizard.

| Name          | Description   |
|---------------|---|
| Time          | DATE property that holds a COM DATE (m_time)  |
| Figure        | Indexed VARIANT property for the four figures on the clock face (m_strFigure[])     |
| PofrochWin    | Method that invalidates the view window and brings the main frame window to the top |
| Kellesiiwiii  | (Refresh)   |
| ShowWin       | Method that displays the application's main window (ShowWin)                        |
| Greatellarm   | Method that creates a CAlarm object and returns its IDispatch pointer               |
| CIEACEAIAIIII | (CreateAlarm)   |

# Table 2.

Launch ClassWizard and click the Automation tab, select CMymfc29CDoc in Class name combo box and click Add Property button and fill in the information as shown in the following Figures.

| MFC ClassWizard  |   | ? 🗙  |
|--|---|--|
| Message Maps Member Variables<br>Project:<br>mymfc29C<br>F:\\mymfc29C\mymfc29CDoc.h, F:\\my<br>External names: | Automation       ActiveX Events       Class Info         Class name:       CMymfc29CD oc       Image: Ima | Add Class ▼<br><u>A</u> dd Method<br>Add Property<br><u>D</u> elete<br><u>E</u> dit Code<br>Data <u>B</u> inding |
| Implementation:  | <br><br>ОК  | Cancel   |

Figure 25: Adding property through Automation page of ClassWizard.

Fill up the property information as shown in the following Figures.

| Add Property            |                         |                   | ? 🔀    |
|-------------------------|-------------------------|-------------------|--------|
| <u>E</u> xternal name:  | Time                    | •                 | OK     |
| <u>T</u> ype:           | DATE                    | <b>•</b>          | Cancel |
| Variable <u>n</u> ame:  | m_time                  |                   |        |
| Notification function:  | OnTimeChanged           |                   |        |
| Implementation          |                         |                   |        |
| C Stock G               | Member <u>v</u> ariable | C Get/Set methods |        |
| <u>P</u> arameter list: |                         |                   |        |
| Name                    | Туре                    |                   |        |
|                         |                         |                   |        |
|                         |                         |                   |        |
|                         |                         |                   |        |

| Figure | 26: | Time | property. |
|--------|-----|------|-----------|
|--------|-----|------|-----------|

| Add Property  |  | ? 🛛          |
|---|--|--------------|
| External name:<br><u>I</u> ype:<br><u>G</u> et function:<br><u>S</u> et function:<br>Implementation-<br>C Stock | Figure  ▼ VARIANT  GetFigure SetFigure O Member ⊻ariable • Get/Set methods | OK<br>Cancel |
| <u>P</u> arameter list:<br><b>Name</b><br>n   | Type<br>short  |              |

Figure 27: Figure property.

Click the Add Method button and add methods as shown in the following Figures.

| Add Method                           |          | ? 🔀 |
|--------------------------------------|----------|-----|
| <u>E</u> xternal name:<br>RefreshWin | •        | OK  |
| Internal <u>n</u> ame:               |          |     |
| Refresh                              |          |     |
| Return <u>t</u> ype:                 |          |     |
| void                                 | <b>•</b> |     |
| -Implementation-                     |          |     |
| C <u>S</u> tock                      | Custom   |     |
| <u>P</u> arameter list:              |          |     |
| Name                                 | Туре     |     |
|                                      |          |     |
|                                      |          |     |
|                                      |          |     |
|                                      |          |     |

Figure 28: RefreshWin method.

| Add Method   |        | ? 🛛          |
|--|--------|--------------|
| External name:<br>ShowWin<br>Internal <u>n</u> ame:<br>ShowWin<br>Return <u>type:</u><br>void<br>Implementation<br>Stock | Custom | OK<br>Cancel |
| <u>P</u> arameter list:  |        |              |
| Name   | Туре   |              |
|  |        |              |
|  |        |              |
| 1  |        |              |

Figure 29: ShowWin method.

| Add Method                    |          | ? 🛛      |
|-------------------------------|----------|----------|
| External name:<br>CreateAlarm | •        | ОК       |
| ,<br>Internal <u>n</u> ame:   |          | Cancel   |
| CreateAlarm                   |          |          |
| Return <u>t</u> ype:          |          |          |
| LPDISPATCH                    | -        |          |
| - Implementation-             |          |          |
| C <u>S</u> tock               | C Lustom |          |
| <u>P</u> arameter list:       |          |          |
| Name                          | Туре     |          |
| time                          | DATE     | <b>•</b> |
|                               |          |          |
|                               |          |          |
|                               |          |          |

Figure 30: CreateAlarm method.

| MFC ClassWizard   |   | ? 🛛  |
|---|---|--|
| Message Maps       Member Variables         Project:         mymfc29C         F:\\mymfc29C\mymfc29CDoc.h, F:\\my         External names:         M         CreateAlarm         C         Figure         M         RefreshWin         M         ShowWin         C         Time | Automation       ActiveX Events       Class Info         Class name:       CMymfc29CDoc       Image: Imag | Add Class  Add Method Add Property Delete Edit Code Data Binding |
| Implementation:<br>LPDISPATCH CreateAlarm(DATE time);<br>Default property   | 0   | < Cancel   |

Figure 31: The added properties and methods to CMymfc29Doc class.

Add the SaveModified() message handler to CMymfc29CDoc class.

| MFC ClassWizard  | ]                |  |            | ? 🛛   |
|--|------------------|--|------------|---|
| Message Maps Project: mymfc29C F:\\mymfc29C\my Object IDs: CMymfc29CDoc ID_APP_ABOUT ID_APP_EXIT ID_EDIT_COPY ID_EDIT_CUT ID_EDIT_CUT ID_EDIT_UNDO Member functions: V_OnNewDocum V_SaveModified V_Serialize Description: Ca | Member Variables | Automation ActiveX Events Class name:<br>Class name:<br>CMymfc29CDoc<br>wmfc29CDoc.cpp<br>Messages:<br>OnFinalRelease<br>OnNewDocument<br>OnOpenDocument<br>OnSaveDocument<br>ReportSaveLoadException<br>SaveModified<br>Serialize | Class Info | Add Class  Add Function Delete Function Edit Code |
|  |                  |  | 01         | Cancel  |

Figure 32: Adding SaveModified() message handler to CMymfc29CDoc class.

#### CMymfc29CDoc class

Add public member variables for the mymfc29CDoc class in the mymfc29CDoc.h.

```
public:
        CString m_strFigure[4];
        CAlarm * m_pAlarm;
class CMymfc29CDoc : public CDocument
{
public:
        CString m_strFigure[4];
        CAlarm * m_pAlarm;
protected: // create from serializatio
```

Listing 5.

Edit/add codes for methods, properties and message handlers of the CMymfc29C1Doc class as shown in the following.

```
CMymfc29CDoc::CMymfc29CDoc()
{
       // TODO: add one-time construction code here
      TRACE("Entering CMymfc29CDoc ctor\n");
      m_time = COleDateTime(0, 0, 0, 5, 10, 15);
       // bogus initial values
      m_strFigure[0] = "XII"; m_strFigure[1] = "III";
      m_strFigure[2] = "VI"; m_strFigure[3] = "IX";
      m_pAlarm = NULL;
      EnableAutomation();
      AfxOleLockApp();
}
// CMymfc29CDoc construction/destruction
CMymfc29CDoc::CMymfc29CDoc()
₹.
          // TODO: add one-time construction code here
     TRACE("Entering CMymfc29CDoc ctor\n");
     m_time = COleDateTime(0, 0, 0, 5, 10, 15);
    // bogus initial values
m_strFigure[0] = "XII"; m_strFigure[1] = "III";
m_strFigure[2] = "VI"; m_strFigure[3] = "IX";
     m_pAlarm = ŇUĹL;
     EnableAutomation();
     AfxOleLockApp();
}
                                        Listing 6.
```

```
VARIANT CMymfc29CDoc::GetFigure(short n)
{
    TRACE("Entering CMymfc29CDoc::GetFigure -- n = %d m_strFigure[n] =
%s\n",
    n, m_strFigure[n]);
    return COleVariant(m_strFigure[n]).Detach();
```

}

Listing 7.

```
void CMymfc29CDoc::SetFigure(short n, const VARIANT FAR& newValue)
{
      // TODO: Add your property handler here
      TRACE("Entering CMymfc29CDoc::SetFigure -n = d, vt = d n, n,
               newValue.vt);
      COleVariant vaTemp;
      vaTemp.ChangeType(VT_BSTR, (COleVariant*) &newValue);
      m_strFigure[n] = vaTemp.bstrVal; // converts double-to-single
}
void CMymfc29CDoc::SetFigure(short n, const VARIANT FAR& newValue)
ł
     // TODO: Add your property handler here
    TRACE("Entering CMymfc29CDoc::SetFigure -- n = %d, vt = %d\n", n,
           newValue.vt);
    COleVariant vaTemp.
    vaTemp.ChangeType(VT_BSTR, (COleVariant*) &newValue);
    m_strFigure[n] = vaTemp.bstrVal; // converts double-to-single
}
                                    Listing 8.
void CMymfc29CDoc::ShowWin()
{
      // TODO: Add your dispatch handler code here
      TRACE("Entering CMymfc29CDoc::ShowWin\n");
      CRect rectWindow;
      CWnd* pFrm = AfxGetApp()->m_pMainWnd;
      pFrm->GetWindowRect(rectWindow);
      WINDOWPLACEMENT wndpl;
      wndpl.length = sizeof(WINDOWPLACEMENT);
      wndpl.showCmd = SW SHOWNORMAL;
      wndpl.rcNormalPosition.left = rectWindow.left;
      wndpl.rcNormalPosition.top = rectWindow.top;
```

wndpl.rcNormalPosition.right = rectWindow.left + 150; wndpl.rcNormalPosition.bottom = rectWindow.top + 150;

pFrm->SetWindowPlacement(&wndpl);

pFrm->UpdateWindow();
pFrm->BringWindowToTop();

}

pFrm->ShowWindow(AfxGetApp()->m\_nCmdShow);

```
void CMymfc29CDoc::ShowWin()
{
    // TODO: Add your dispatch handler code here
    TRACE("Entering CMymfc29CDoc::ShowWin\n");
    CRect rectWindow;
    CWnd* pFrm = AfxGetApp()->m_pMainWnd;
    pFrm->GetWindowRect(rectWindow);
    WINDOWPLACEMENT wndpl;
    wndpl.length = sizeof(WINDOWPLACEMENT);
    wndpl.showCmd = SW_SHOWNORMAL;
    wndpl.rcNormalPosition.left = rectWindow.left;
    wndpl.rcNormalPosition.top = rectWindow.top;
    wndpl.rcNormalPosition.bottom = rectWindow.top + 150;
    wndpl.rcNormalPosition.bottom = rectWindow.top + 150;
    pFrm->SetWindow(AfxGetApp()->m_nCmdShow);
    pFrm->BringWindowToTop();
```

```
}
```

Listing 9.

```
LPDISPATCH CMymfc29CDoc::CreateAlarm(DATE time)
{
      // TODO: Add your dispatch handler code here
      TRACE("Entering CMymfc29CDoc::CreateAlarm, time = %f\n", time);
      // OLE deletes any prior CAlarm object
      m_pAlarm = new CAlarm(time);
      return m_pAlarm->GetIDispatch(FALSE); // no AddRef here
}
BOOL CMymfc29CDoc::SaveModified()
ł
      // TODO: Add your specialized code here and/or call the base class
      return TRUE;
}
LPDISPATCH CMymfc29CDoc::CreateAlarm(DATE time)
 ł
     7 TODO: Add your dispatch handler code here
    TRACE("Entering CMymfc29CDoc::CreateAlarm, time = %f\n", time);
     // OLE deletes any prior CAlarm object
     m_pAlarm = new CAlarm(time);
     return m_pAlarm->GetIDispatch(FALSE);
                                            // no AddRef here
 }
BOOL CMymfc29CDoc::SaveModified()
 {
     // TODO: Add your specialized code here and/or call the base class
    return TRUE;
 }
```

Listing 10.

#### CMainFrame class

Add the following statement in the CMainFrame::PreCreateWindow function to disable the menu.

```
cs.hMenu = NULL; // no menu
return CFrameWnd::PreCreateWindow(cs);
```

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.hMenu = NULL; // no menu
    return CFrameWnd::PreCreateWindow(cs);
}
```

Listing 11.

#### CMymfc29CView class

Using ClassView or manually add the following data members in mymfc29CView.h.

```
private:
    static const int nX[60];
    static const int nY[60];
    CFont m_font;
class CMymfc29CView : public CView
{
    private:
        static const int nX[60];
        static const int nY[60];
        CFont m_font;
```

Listing 12.

Add the following in the **mymfc29CView.cpp**.

```
// computer-generated x-y coordinates for 60 points around a circle
const int CMymfc29CView::nX[] = {
          0, 105, 208, 309, 407, 500, 588, 669, 743, 809, 866, 914,
          951, 978, 995, 1000, 995, 978, 951, 914, 866, 809, 743, 669,
          588, 500, 407, 309, 208, 105, 0, -104, -207, -308, -406, -499,
          -587, -668, -742, -808, -865, -913, -950, -977, -994, -999, -994, -977,
         -950, -913, -865, -808, -742, -668, -587, -499, -406, -308, -207, -104 };
const int CMymfc29CView::nY[] = {
         999, 994, 978, 951, 913, 866, 809, 743, 669, 587, 499, 406,
         309, 207, 104, 0, -104, -207, -309, -406, -499, -587, -669, -743,
         -809, -866, -913, -951, -978, -994, -999, -994, -978, -951, -913, -866,
         -809, -743, -669, -587, -500, -406, -309, -207, -104, 0, 104, 207,
          309, 406, 499, 587, 669, 743, 809, 866, 913, 951, 978, 994 };
// CMymfc29CView
 // computer-generated x-y coordinates for 60 points around a circle
 const int CMymfc29CView::nX[] =
          \begin{array}{c} 180 \\ 0, \ 105, \ 208, \ 309, \ 407, \ 500, \ 588, \ 669, \ 743, \ 809, \ 866, \ 914, \\ 951, \ 978, \ 995, \ 1000, \ 995, \ 978, \ 951, \ 914, \ 866, \ 809, \ 743, \ 669 \\ 588, \ 500, \ 407, \ 309, \ 208, \ 105, \ 0, \ -104, \ -207, \ -308, \ -406, \ -4 \\ 588, \ 500, \ 407, \ 309, \ 208, \ 105, \ 0, \ -104, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \ 500, \
581, 570, 593, 1000, 593, 570, 581, 514, 500, 500, 743, 605,
588, 500, 407, 309, 208, 105, 0, -104, -207, -308, -406, -499,
-587, -668, -742, -808, -865, -913, -950, -977, -994, -999, -994, -977,
-950, -913, -865, -808, -742, -668, -587, -499, -406, -308, -207, -104 };
const int CMymfc29CView::nY[] = {
999, 994, 978, 951, 913, 866, 809, 743, 669, 587, 499, 406,
209, 207, 104, 0, 104, 207, 209, 406, 499, 587, 499, 406,
          309, 207, 104, 0, -104, -207, -309, -406, -499, -587, -669, -743,
-809, -866, -913, -951, -978, -994, -999, -994, -978, -951, -913,
-809, -743, -669, -587, -500, -406, -309, -207, -104, 0, 104, 207,
309, 406, 499, 587, 669, 743, 809, 866, 913, 951, 978, 994 };
                                                                                                                                                                                                                               -866.
 IMPLEMENT_DYNCREATE(CMymfc29CView, CView)
```

Listing 13.

Add the code for the CMymfc29CView::OnDraw() member function.

```
void CMymfc29CView::OnDraw(CDC* pDC)
{
   CMymfc29CDoc* pDoc = GetDocument();
   ASSERT_VALID(pDoc);
   CRect rectClient;
   CPen pen0(PS_SOLID, 0, (COLORREF) 0);
   CPen pen1(PS_SOLID, 20, (COLORREF) 0);
   CPen pen2(PS_SOLID, 30, (COLORREF) 0);
   int i, j, n, m, nS, nM, nH;
   CSize sz;
   pDC->SetMapMode(MM_ISOTROPIC);
   GetClientRect(rectClient);
   pDC->SetWindowExt(2400, 2400);
   pDC->SetViewportExt(rectClient.right, -rectClient.bottom);
   pDC->SetViewportOrg(rectClient.right / 2, rectClient.bottom / 2);
   pDC->SelectObject(&m_font);
  pDC->SelectObject(&pen0);
   for(i = 0; i < 60; i++) { // second/minute ticks</pre>
      pDC->MoveTo(nX[i], nY[i]);
      pDC->LineTo(nX[i] * 9 / 10, nY[i] * 9 / 10);
   }
   pDC->SelectObject(&pen1);
   for(i = 0; i < 12; i++) { // hour ticks</pre>
      j = i * 5;
      pDC->MoveTo(nX[j], nY[j]);
      pDC->LineTo(nX[j] * 8 / 10, nY[j] * 8 / 10);
   for(i = 0; i < 4; i++) { // figures</pre>
      j = i * 15;
      sz = pDC->GetTextExtent(pDoc->m_strFigure[i], pDoc-
>m_strFigure[i].GetLength());
      n = nX[j] * 10 / 9 - sz.cx / 2;
      m = nY[j] * 10 / 9 + sz.cy / 2;
      pDC->TextOut(n, m, pDoc->m_strFigure[i]);
   }
   // draw the clock hands
   COleDateTime dt = pDoc->m_time;
   nH = dt.GetHour();
  nM = dt.GetMinute();
  nS = dt.GetSecond();
  pDC->SelectObject(&pen0);
   i = nS;
  pDC->MoveTo(0, 0);
  pDC->LineTo(nX[i], nY[i]); // seconds
  pDC->SelectObject(&pen1);
   i = nM;
  pDC->MoveTo(0, 0);
  pDC->LineTo(nX[i] * 8 / 10, nY[i] * 8 / 10); // minutes
  pDC->SelectObject(&pen2);
   i = nH * 5;
   i = (i % 60) + nM / 12;
  pDC->MoveTo(0, 0);
  pDC->LineTo(nX[i] * 6 / 10, nY[i] * 6 / 10); // hours
  pDC->SelectStockObject(BLACK_PEN);
   CAlarm* pAlarm;
   if((pAlarm = pDoc->m_pAlarm) != NULL) {
```

```
COleDateTime dt = pAlarm->m_time;
nH = dt.GetHour();
nM = dt.GetMinute();
nS = dt.GetSecond();
TRACE("dt = %f, nH = %d, nM = %d, nS = %d\n", dt.m_dt, nH, nM, nS);
i = nH * 5;
i = (i % 60) + nM / 12;
pDC->TextOut(nX[i] * 8 / 10, nY[i] * 8 / 10, "A"); // alarm
}
```

Add the WM\_CREATE Windows message handler.

| MFC ClassWizar   | I   |   |            | ? 🗙                                |  |  |  |  |
|--|---|---|------------|------------------------------------|--|--|--|--|
| MFC ClassWizard<br>Message Maps<br>Project:<br>mymfc29C<br>F:\\mymfc29C\my<br>Object IDs:<br>CMymfc29CView<br>ID_APP_APOUT | Member Variables   A<br>vmfc29CView.h, F:\\my                   | Automation ActiveX Events Class <u>n</u> ame:<br>Class <u>n</u> ame:<br>CMymfc29CView<br>mfc29CView.cpp<br>Messages:<br>WM_CANCELMODE | Class Info | Add Class  Add Function  Edit Code |  |  |  |  |
| ID_APP_ABUUT<br>ID_APP_EXIT<br>ID_EDIT_COPY<br>ID_EDIT_CUT<br>ID_EDIT_PASTE<br>ID_EDIT_UNDO<br>Member functions:           | ■   | WM_CAPTURECHANGED<br>WM_CHAR<br>WM_CONTEXTMENU<br>WM_COPYDATA<br>WM_CREATE<br>WM_DESTROY  |            | <u>(</u>                           |  |  |  |  |
| W OnCreate<br>V OnDraw<br>V PreCreateWin   | ON_WM_C   | REATE   |            |                                    |  |  |  |  |
| Description: In  | Description: Indicates that a window is being created OK Cancel |   |            |                                    |  |  |  |  |

Figure 33: Adding WM\_CREATE Windows message handler to CMymfc29CView class.

Then, add/edit the OnCreate() code.

Listing 14.

Add the **#include** statement for the **mymfc29CView.cpp**.

#include "Alarm.h"
#include "stdafx.h"
#include "mymfc29C.h"
#include "Alarm.h"
#include "mymfc29CDoc.h"
#include "mymfc29CView.h"

Listing 15.

Change the protected OLE dispatch map functions in **mymfc29CDoc.h** to public (or you can use friend keyword).

```
public:
    // Generated OLE dispatch map functions
    //{{AFX_DISPATCH(CMymfc29CDoc)
    DATE m_time;
    afx_msg void OnTimeChanged();
    afx_msg void Refresh();
    afx_msg void ShowWin();
    afx_msg IPDISPATCH CreateAlarm(DATE time);
    afx_msg VARIANT GetFigure(short n);
    afx_msg void SetFigure(short n, const VARIANT FAR& newValue);
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};
```

Listing 16.

Also change the protected OLE dispatch map functions in **Alarm.h** to public (or you can use friend keyword).

```
public:
    //{{AFX_DISPATCH(CAlarm)
    DATE m_time;
    afx_msg void OnTimeChanged();
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};
```

Listing 17.

Add the **#include** statement to the **mymfc29C.cpp** and **mymfc29CDoc.cpp** file.

```
#include "Alarm.h"
#include "stdafx.h"
#include "MainFrm.h"
#include "MainFrm.h"
#include "mymfc29CDoc.h"
#include "mymfc29CView.h"
Listing 18.
#include "stdafx.h"
#include "Alarm.h"
#include "Alarm.h"
#include "Alarm.h"
#include "mymfc29CDoc.h"
```

### Listing 19.

Build the program and make sure there is no error. Next, we need a client to test our component.

For the Excel, launch Excel, save the file as <u>mymfc29C.xls</u> and invoke the Visual Basic editor. Double-click the **sheet1** on the top-left window. Copy and paste the following macros.

```
Dim Clock As Object
Dim Alarm As Object
Sub LoadClock()
    Set Clock = CreateObject("Mymfc29C.Document")
   Range("A3").Select
   n = 0
    Do Until n < 4
        Clock.figure(n) = Selection.Value
        Selection.Offset(0, 1).Range("A1").Select
        n = n + 1
    Loop
   RefreshClock
    Clock.ShowWin
End Sub
Sub RefreshClock()
    Clock.Time = Now()
    Clock.RefreshWin
End Sub
Sub CreateAlarm()
    Range("E3").Select
    Set Alarm = Clock.CreateAlarm(Selection.Value)
    RefreshClock
End Sub
Sub UnloadClock()
    Set Clock = Nothing
End Sub
```

| 着 Microsoft Visual Bas     | - mymfc29C.xls - [Sheet1 (Code)]  |          |
|----------------------------|---|----------|
| 🗄 🖾 Eile Edit View I       | ert F <u>o</u> rmat <u>D</u> ebug <u>R</u> un <u>T</u> ools <u>A</u> dd-Ins <u>W</u> indow <u>H</u> elp | 8×       |
| i 🗙 🤚 - 🕞   X 📭 I          | LAAI 🤊 (° I 🕨 u u 🖬 🕍 St 😤 😤 🛞 🞯  | 1        |
| Project - VBAProject 🔀     | (General)  LoadClock  | Ţ        |
|                            | Dim Cleat la Chiest   | -        |
| AcrobatPDFMaker (PDF       | Dim Alarm As Object   | -        |
| VBAProject (mymfc29C       |   |          |
| 📇 Microsoft Excel Objects  | Sub LoadClock()   |          |
| Breet1 (Sheet1)            | Set Clock = CreateObject("Mymfc29C.Document"<br>Denge("M3") Select                                      | )        |
| Breet3 (Sheet3)            | n = 0   |          |
| ThisWorkbook               | Do Until n < 4  |          |
|                            | Clock.figure(n) = Selection.Value   |          |
|                            | Selection.Offset(0, 1).Range("A1").Selec  | t        |
| Properties - Sheet1 🗙      | Loop  |          |
| Sheet1 Worksheet 🔹         | RefreshClock  |          |
| Alphabetic Categorized     | Clock.ShowWin   |          |
| (Name) Sheet1              | End Sub   | -        |
| DisplayPageB False         | Sub RefreshClock()  |          |
| DisplayRightTFalse         | Clock.Time = Now()  |          |
| EnableCalcula True         | Clock.RefreshWin  |          |
| EnableOutlinir False       | End Sub   | _        |
| EnablePivotT{False         | Sub CreateAlarm()   |          |
| EnableSelecti(0 - xINoRest | Range("E3").Select  |          |
| ScrollArea                 | Set Alarm = Clock.CreateAlarm(Selection.Valu  | .e)      |
| StandardWidt 8.43          | End Sub   |          |
| Visible -1 - xlSheet'      |   | -        |
|                            | Sub UnloadClock()   |          |
|                            | Set Clock = Nothing   | _        |
|                            |   |          |
| <u> </u>                   |   | <u> </u> |

Figure 34: Creating macros with Visual Basic editor.

Arrange the buttons as shown below and assign the previously created macros to the buttons appropriately. Be careful, the cells in macro code should match with the Excel cell.

| <b>N</b> | licro | soft Ex      | cel -   | mymfc2 | 9C.xls          |               |              |                |              |                |         |       |     |
|----------|-------|--------------|---------|--------|-----------------|---------------|--------------|----------------|--------------|----------------|---------|-------|-----|
| :2       | Eile  | <u>E</u> dit | ⊻iew    | Insert | F <u>o</u> rmat | <u>T</u> ools | <u>D</u> ata | <u>W</u> indow | <u>H</u> elp | Ado <u>b</u> e | PDF     | -     | ₽×  |
| : 🗅      | 2     | J 🔒          |         | 🕘   🛍  | - 🖺             | 10 -          | 🧕 Σ          | - Ag↓   [      | 1 0          | ••<br>₹        | Arial   |       | • " |
| : 🔁      |       | 🐮 🖕          | Aa      | ab 🏾 🖄 |                 | •             |              |                |              |                | J   📰 1 | Ţ     |     |
|          | E3    |              | -       | fx     |                 |               |              |                |              |                |         |       |     |
|          |       | A            |         | В      | C               |               | D            | E              |              | F              | G       | Н     | ~   |
| 1        | - L   | load Cl      | lock    |        | Refresh         | Time          |              | Set            | Alarm        |                | Unload  | Clock | ┣┛  |
| 3        |       |              |         |        |                 |               |              |                |              |                |         |       |     |
| 4        |       |              |         |        |                 |               |              |                |              |                |         |       | _   |
| 5        |       |              | ot1 /   | Choot? | / Choots        |               |              |                |              |                | ļ)      |       |     |
| 14 4     |       | -  \Sne      | set i A | Sneetz | X sneets        | <u> </u>      |              |                |              |                |         |       |     |
| Read     | У     |              |         |        |                 |               |              |                |              |                |         |       | ;;  |

Figure 35: mymfc29C.xls button layout.

Next, we are going to test the MYMFC29C component. Click the Load Clock button.

| <b>X</b> N | 🛛 Microsoft Excel - mymfc29C.xls |              |              |         |                 |               |              |                |              |                  |         |       |     |
|------------|----------------------------------|--------------|--------------|---------|-----------------|---------------|--------------|----------------|--------------|------------------|---------|-------|-----|
| :          | Eile                             | <u>E</u> dit | <u>V</u> iew | Insert  | F <u>o</u> rmat | <u>T</u> ools | <u>D</u> ata | <u>W</u> indow | <u>H</u> elp | Ado <u>b</u> e I | PDF     | -     | 8×  |
| : 🗅        | 2                                |              |              | 3 🛍     | - 🖺 -           | 10 -          | 🧕 Σ          | - AZ↓          | 11 🤇         |                  | Arial   |       | • " |
| : 🔁        |                                  | も 🖕          | Aa           | ab 🛛 🖆  |                 | •             |              |                | 1            | 1                | I   📰 1 | Ţ     |     |
|            | E3                               |              | •            | fx      | 10:00:00        | AM            |              |                |              |                  |         |       |     |
|            |                                  | А            |              | В       | C               |               | D            | E              |              | F                | G       | H     | _   |
| 1          | - l                              | Load C       | lock         |         | Refresh         | Time          |              | Set            | Alarn        | ۱ —              | Unload  | Clock | HĒ  |
| 3          | XII                              |              | III          |         | VI              | DX            | (            | 10:00 /        | AM           |                  |         |       |     |
| 4          |                                  |              |              |         |                 |               |              |                |              |                  |         |       | _   |
| 5          |                                  |              |              |         |                 |               |              |                |              |                  |         |       |     |
| 6          |                                  |              |              |         |                 |               |              |                |              |                  |         |       | -   |
| 7<br>      |                                  | • ∖She       | eet1 /       | (Sheet2 | / Sheet3        | 1             |              | <              |              |                  | ]       |       | Σ   |
| Read       | ły                               |              |              |         |                 |               |              | -              |              |                  |         |       |     |

Figure 36: mymfc29C.xls, a client program in action.

The following clock is launched. At the same time XII, III, VI and IX value will be displayed in the appropriate Excel cells. Type the alarm in **E3** cell and click the **Set Alarm** button. The alarm will be set in the clock as shown in Figure 38.



Figure 37: Clock (component) triggered through client program (Excel).



Figure 38: Alarm was set through a client.



Figure 39: MYMFC29C in action, component and client side-by-side.

# The Story

If you're wondering why there's no menu, it's because of the following statement in the CMainFrame::PreCreateWindow function:

```
cs.hMenu = NULL; // no menu
return CFrameWnd::PreCreateWindow(cs);
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.hMenu = NULL; // no menu
return CFrameWnd::PreCreateWindow(cs);
}
```

Listing 20.

Close the **Clock** program and then click the **Unload Clock** button. If you've started the component program from the debugger, you can watch the Debug window for a message box that indicates that the ExitInstance() function is called.

AppWizard did most of the work of setting up the document as an Automation component. In the derived application class CMymfc29CApp, it generated a data member for the component, as shown here:

```
public:
    COleTemplateServer m_server;
```

The MFC COleTemplateServer class is derived from COleObjectFactory. It is designed to create a COM document object when a client calls IClassFactory::CreateInstance. The class ID comes from the **global clsid** variable defined in **mymfc29C.cpp**. The human-readable program ID (**Mymfc29C.Document**) comes from the IDR\_MAINFRAME string resource.

In the InitInstance() function (in **mymfc29C.cpp**), AppWizard generated the code below, which connects the component object (the document) to the application's document template.

```
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CMymfc29CDoc),
        RUNTIME_CLASS(CMainFrame), // main SDI frame window
        RUNTIME_CLASS(CMymfc29CView));
AddDocTemplate(pDocTemplate);
.
.
.
.
m_server.ConnectTemplate(clsid, pDocTemplate, TRUE);
```

Now all the plumbing is in place for COM and the framework to construct the document, together with the view and frame. When the objects are constructed, however, the main window is not made visible. That's your job. You must write a method that shows the window.

The following UpdateRegistry() call from the InitInstance() function updates the Windows Registry with the contents of the project's IDR\_MAINFRAME string resource:

```
m_server.UpdateRegistry(OAT_DISPATCH_OBJECT);
```

The following dispatch map in the **mymfc29CDoc.cpp** file shows the properties and methods for the CMymfc29CDoc class. Note that the Figure property is an indexed property that ClassWizard can generate if you specify a parameter. Later you'll see the code you have to write for the GetFigure() and SetFigure() functions.

```
BEGIN_DISPATCH_MAP(CMymfc29CDoc, CDocument)
    //{{AFX_DISPATCH_MAP(CMymfc29CDoc)
    DISP_PROPERTY_NOTIFY(CMymfc29CDoc, "Time", m_time, OnTimeChanged,
        VT_DATE)
    DISP_FUNCTION(CMymfc29CDoc, "ShowWin", ShowWin, VT_EMPTY, VTS_NONE)
    DISP_FUNCTION(CMymfc29CDoc, "CreateAlarm", CreateAlarm,
        VT_DISPATCH, VTS_DATE)
    DISP_FUNCTION(CMymfc29CDoc, "RefreshWin", Refresh, VT_EMPTY,
        VTS_NONE)
    DISP_PROPERTY_PARAM(CMymfc29CDoc, "Figure", GetFigure, SetFigure,
        VT_VARIANT, VTS_I2)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()
...
```

The ShowWin() and RefreshWin() member functions aren't very interesting, but the CreateAlarm() method is worth a close look. Here's the corresponding CreateAlarm() member function:

```
LPDISPATCH CMymfc29CDoc::CreateAlarm(DATE time)
{
    TRACE("Entering CMymfc29CDoc::CreateAlarm, time = %f\n", time);
    // OLE deletes any prior CAlarm object
    m_pAlarm = new CAlarm(time);
    return m_pAlarm->GetIDispatch(FALSE); // no AddRef() here
```

}

We've chosen to have the component create an alarm object when a controller calls CreateAlarm(). CAlarm is an **Automation component** class that we've generated with ClassWizard. It is not COM-creatable, which means there's no IMPLEMENT\_OLECREATE macro and no class factory. The CreateAlarm() function constructs a CAlarm object and returns an IDispatch pointer. The FALSE parameter for CCmdTarget::GetIDispatch means that the reference count is not incremented; the CAlarm object already has a reference count of 1 when it is constructed.

The CAlarm class is declared in Alarm.h as follows:

```
class CAlarm : public CCmdTarget
    DECLARE DYNAMIC(CAlarm)
public:
    CAlarm(DATE time);
// Attributes
public:
// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAlarm)
    public:
    virtual void OnFinalRelease();
    //}}AFX_VIRTUAL
// Implementation
protected:
    virtual ~CAlarm();
    // Generated message map functions
    //{{AFX_MSG(CAlarm)
        // NOTE - the ClassWizard will add and remove member
        // functions here.
    //} AFX_MSG
    DECLARE MESSAGE MAP()
    // Generated OLE dispatch map functions
public:
    //{{AFX_DISPATCH(CAlarm)
    DATE m_time;
    //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};
```

Notice the absence of the DECLARE\_DYNCREATE macro. Alarm.cpp contains a dispatch map, as follows:

```
BEGIN_DISPATCH_MAP(CAlarm, CCmdTarget)
    //{{AFX_DISPATCH_MAP(CAlarm)
    DISP_PROPERTY(CAlarm, "Time", m_time, VT_DATE)
    //}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()
```

Why do we have a CAlarm class? We could have added an AlarmTime property in the CMymfc29CDoc class instead, but then we would have needed another property or method to turn the alarm on and off. By using the CAlarm class, what we're really doing is setting ourselves up to support multiple alarms, a collection of alarms.

To implement an **Automation collection**, we would write another class, CAlarms, which would contain the methods Add, Remove, and Item. Add and Remove are self-explanatory; Item returns an IDispatch pointer for a collection element identified by an index, numeric, or some other key. We would also implement a read-only Count property that returned the number of elements. The document class (which owns the collection) would have an Alarms method with an optional VARIANT parameter. If the parameter were omitted, the method would return the IDispatch pointer for the collection. If the parameter specified an index, the method would return an IDispatch pointer for the selected alarm.

If we wanted our collection to support the VBA "For Each" syntax, we'd have some more work to do. We'd add an IEnum VARIANT interface to the CAlarms class to enumerate the collection of variants and implement the Next() member function of this interface to step through the collection. Then we'd add a CAlarms method named \_NewEnum that returned an IEnumVARIANT interface pointer. If we wanted the collection to be general, we'd allow separate enumerator objects (with an IEnumVARIANT interface) and we'd implement the other IEnumVARIANT functions, Skip(), Reset(), and Clone().

The Figure property is an indexed property, which makes it interesting. The Figure property represents the four figures on the clock face, XII, III, VI, and IX. It's a CString array, so we can use Roman numerals. Here's the declaration in **mymfc29CDoc.h**:

```
public:
    CString m_strFigure[4];
```

And here are the GetFigure() and SetFigure() functions in mymfc29CDoc.cpp:

These functions tie back to the DISP\_PROPERTY\_PARAM macro in the CMymfc29CDoc dispatch map. The first parameter is the index number, specified as a short integer by the last macro parameter. Property indexes don't have to be integers, and the index can have several components (row and column numbers, for example). The ChangeType() call in SetFigure() is necessary because the controller might otherwise pass numbers instead of strings. You've just seen collection properties and indexed properties. What's the difference? A controller can't add or delete elements of an indexed property, but it can add elements to a collection and it can delete elements from a collection. What draws the clock face? As you might expect, it's the OnDraw() member function of the view class. This function uses GetDocument() to get a pointer to the document object, and then it accesses the document's property data members and method member functions. The Excel macro code appears below.

```
Dim Clock As Object
Dim Alarm As Object
Sub LoadClock()
Set Clock = CreateObject("Mymfc29C.Document")
Range("A3").Select
n = 0
Do Until n < 4
Clock.figure(n) = Selection.Value
Selection.Offset(0, 1).Range("A1").Select
```

```
n = n + 1
    Tioop
    RefreshClock
    Clock.ShowWin
End Sub
Sub RefreshClock()
    Clock.Time = Now()
    Clock.RefreshWin
End Sub
Sub CreateAlarm()
   Range("E3").Select
    Set Alarm = Clock.CreateAlarm(Selection.Value)
    RefreshClock
End Sub
Sub UnloadClock()
    Set Clock = Nothing
End Sub
```

Notice the Set Alarm statement in the CreateAlarm macro. It calls the CreateAlarm method to return an IDispatch pointer, which is stored in an object variable. If the macro is run a second time, a new alarm is created, but the original one is destroyed because its reference count goes to 0.

You've seen a modal dialog in a DLL (MYMFC29B) and you've seen a main frame window in an EXE (MYMFC29C). Be careful with modal dialogs in EXEs. It's fine to have an About dialog that is invoked directly by the component program, but it isn't a good idea to invoke a modal dialog in an out-of-process component method function. The problem is that once the modal dialog is on the screen, the user can switch back to the client program. MFC clients handle this situation with a special "Server Busy" message box, which appears right away. Excel does something similar, but it waits 30 seconds, and this could confuse the user.

Continue on next module. part 3...

-----End Automation part 2-----

# Further reading and digging:

- 1. MSDN MFC 6.0 class library online documentation used throughout this Tutorial.
- 2. MSDN MFC 7.0 class library online documentation used in .Net framework and also backward compatible with 6.0 class library
- 3. MSDN Library
- 4. DCOM at MSDN.
- 5. <u>COM+</u> at MSDN.
- 6. <u>COM</u> at MSDN.
- 7. Windows data type.
- 8. Win32 programming Tutorial.
- 9. The best of C/C++, MFC, Windows and other related books.
- 10. Unicode and Multibyte character set: Story and program examples.