MapPro71.OCX Rel. 2 - Using it with in MS Visual Studio 6.0 C++

Disclaimer: When the SDK is purchased/licensed by a user, it is expected that the user is familiar with their development environment, with the process of importing/adding interfaces from an Active-X controls to their application, with the process of accessing API's from an imported interface, through their application, etc. What follows is a brief outline of how to obtain basic access to the MapPro71.OCX, and register the control for development for users who are just starting out and are not very familiar yet with the IDE they are using. It is **not** a reference document on the general use of MS-VC++, or other IDEs.

***** Prior to being able to use the MapPro71.OCX (just like any other OCX), the control needs to be registered with Windows (using `REGSVR32.EXE` – a Windows program. Check your Windows references if you are not sure how to use it), so that IDEs can be aware of the existence of the OCX on the system)

(A) In Visual Mode
Creating a New Application with a Visual instance of the control

1. Start MS Visual C++
2. Select File, New
3. Type in the project name (e.g. USC1), select MFC AppWizard(Exe) and click O.K. (make sure the Win32 box is checked)
4. Select Dialog Based and Click Finish. The New Project Information Dialog should appear, click O.K. to confirm the creation of the project skeleton.
5. When the form appears, Right-click on it and select Insert Active-X Control
6. Highlight MapPro71 Control (MapPro) in the list and click O.K. The OCX control should appear on the form displaying the version (build) # and other information. Note that if the MapPro71 Control does not appear in the list, it's an indication that the control has not been successfully registered in Windows using RegSVR32.EXE (see comment at the beginning of this document).
7. Right Click on the instance of the control on the form and select Properties. The control's built-in properties dialog should appear.
8. Select the 6th tab from the left (Installation) and type in your Vendor Code and Password. **Make sure** you press Enter, after you type the information, to effect the registration or Development mode and close the dialog.
9. At this point, if everything was done correctly, you should be able to run your application. Select Build, Execute USC1. You should see a map of the USA.
10. Close the running application to return to development mode.

11. Select the MapPro Control, Right Click and select Class wizard
12. Select Member Variables Tab
13. Highlight/Select IDC_MAPPRO1
14. Click Add Variable. You should get a message that the MapPro71 control has not been inserted. Click O.K. for Developer Studio to insert it and create a wrapper for it.
15. The default class name in the next dialog should be CMapPro. Click O.K. to confirm the class and the import process
16. Type in a member variable name e.g. m_mappro and click OK
17. Click OK again. You should see the USC1 classes created on the left side. If you expand CMapPro, you should be able to see the properties and functions. Properties (in C++) are usually imported as a **Get** and **Set** pair.
18. Double-Click on the form to open the USC1Dlg.cpp source file.
19. In the OnInitDialog() routine, and after the comment

```
// TODO: Add extra initialization here
```

Add the following Code:

```
m_mappro.SetToolBarMode(1);
```

20. Select Build, Execute USC1. When the program runs, now, you'll also see the built-in toolbar appear anchored at the top.

(B) In Non-Visual Mode
Creating a New Application with a Dynamically created instance of the control

1. Start MS Visual C++
2. Select File, New
3. Type in a Project name (e.g. USC2), select MFC AppWizard(Exe) and click O.K. (make sure the Win32 box is checked)
4. Select Single Document and Click Finish. The New Project Information Dialog should appear, click O.K.
5. Highlight USC2 Classes (on the left). Select Project, Add to Project, Components and Controls.
6. Select Registered Active X Controls and click Insert.
7. Select MapPro71 Control (MapPro) and click Insert
8. Click O.K. on the next dialog to confirm insertion
9. The default Class name in the next dialog should be CMapPro. Click O.K. to accept it.

10. Close the Components and controls Library dialog.
11. You should see the USC2 classes created on the left side. If you expand CMapPro, you should be able to see the properties and functions. Properties (in C++) are usually imported as a Get and Set pair (e.g., GetDevCode and SetDevCode).
12. Right-click on CMainFrame and select Add Member Variable
13. Type in CMapPro for the Variable Type and MyMap for the Variable Name (make sure Public is checked). Click O.K. to create the variable and close the dialog.
14. Expand the CMainFrame class (on the left) and double click the OnCreate member to go to the code view.
15. In the OnCreate method, you can add the code to register the OCX for development, and any other code that you may want to initialize your application. Here are 4 lines of code that can be used in OnCreate to register the control. (Note that only the last two lines are actually calling MapPro71 APIs, the other lines are C++ system calls – consult your C++ reference material)

```
//-- Dimensions the MapPro71 control as t will be instantiated on the form
    CRect rect(0,20,500,500);
//-- Create an instance of the control, make the application the parent
//-- by using "this". For details on the rest of the arguments, please
//-- consult your C++ reference material
   MyMap.Create(NULL,WS_VISIBLE,rect,this,0,NULL,FALSE);
//-- Set the Development Code and Development Password. If these are
//-- not set correctly, or the process fails, you will get an "Unregistered"
//-- Error dialog when you try to execute your project. Note, you should
//-- substitute your own Vendor Code and Password in your code.
   MyMap.SetDevCode("2xxx-5xxx");
   MyMap.SetDevPass("121212");
```

-- Select the Class Tab (left) and you should see the m_pmap class
-- Source Files and header files should also have the m_pmap files created.

(C) Manually adding methods properties to the Interface

Unfortunately, MS C++ does not automatically create *all* the interface points, as expected (in particular properties marked as invisible). You may have to hand code the Stubs for some of these methods/properties. Note: In addition to that, some of these properties are ReadOnly (e.g. LatCenter), and may not apapear in a property inspector, but should be accessible during run time.

If the c++ class wizard did not import a method it can be added manually given its Dispatch ID. By creating the appropriate stubs in the mappro.ccp and mappro.h

source files. For example, creating the stubs for the property Custom (Dispatch ID 26), which is not automatically imported, is done by

(a) adding this code to the mappro.ccp source file:

```
unsigned long CMapPro::GetCustom()
{
      unsigned long result;
      InvokeHelper(0x1a, DISPATCH_PROPERTYGET, VT_I4,
(void*)&result, NULL);
      return result;
}

void CMapPro::SetCustom(unsigned long newValue)
{
      static BYTE parms[] =
            VTS_I4;
      InvokeHelper(0x1a, DISPATCH_PROPERTYPUT, VT_EMPTY,
NULL, parms,
             newValue);
}
```

(b) adding this code to the mappro.h header file:

```
unsigned long GetCustom();
void SetCustom(unsigned long newValue);
```

(D) Viewing the Interface in MS Visual C++ and Get Dispatch IDs

Select *Tools*, *OLE/COM Object Viewer*, *Type Libraries* and navigate to and select the MapPro71.OCX. This should provide you with the information needed to handcode some of the stubs for certain methods/properties that are not automatically imported by MS-VC++