

# Using IVI-COM Drivers From LabWindows/CVI

## REVISION HISTORY

Date	Description	Revision Letter
11/08/02	Original Draft	A

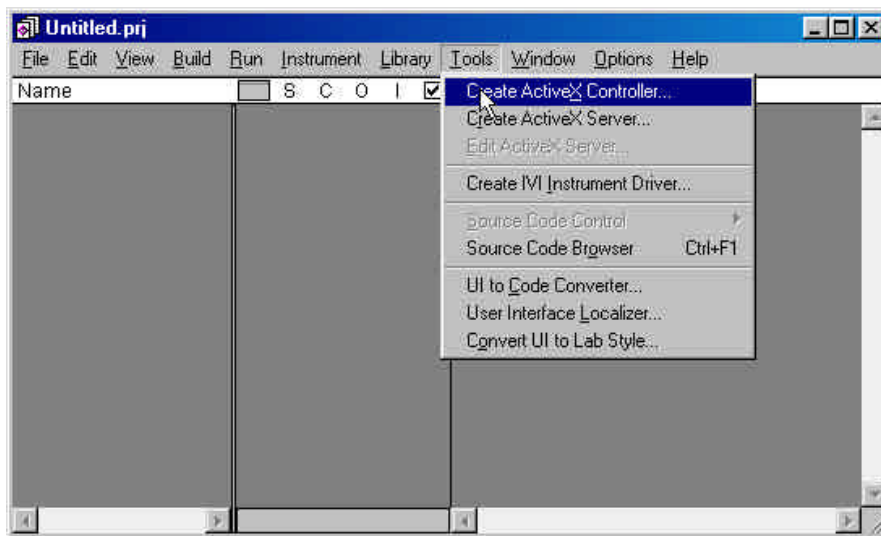
## Abstract

This application note describes the procedure for creating and using a wrapper to allow an IVI-COM driver to be used from LabWindows/CVI environment. The procedure makes use of the “Create ActiveX Controller” feature of the environment, which means that at least version 6.0 of LabWindows/CVI needs to be used.

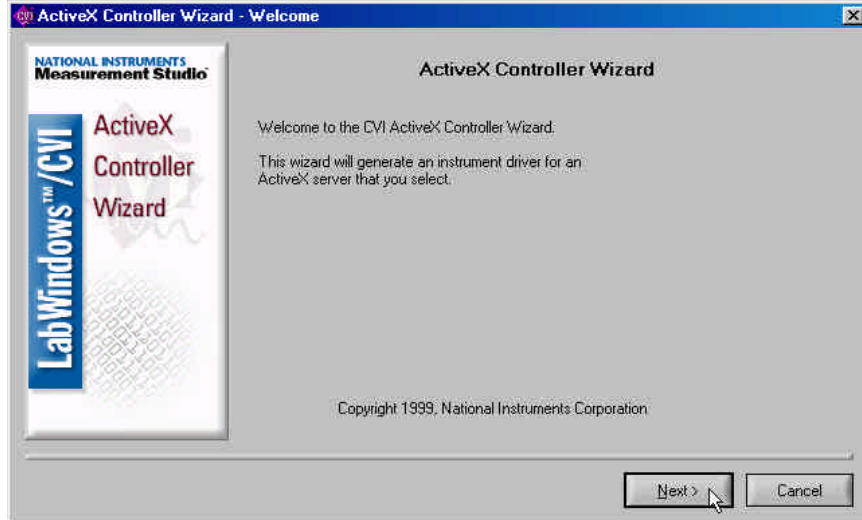
The driver used in the examples in this application note (VektrexScope.dll) is available [here](#).

## Creating the Wrapper

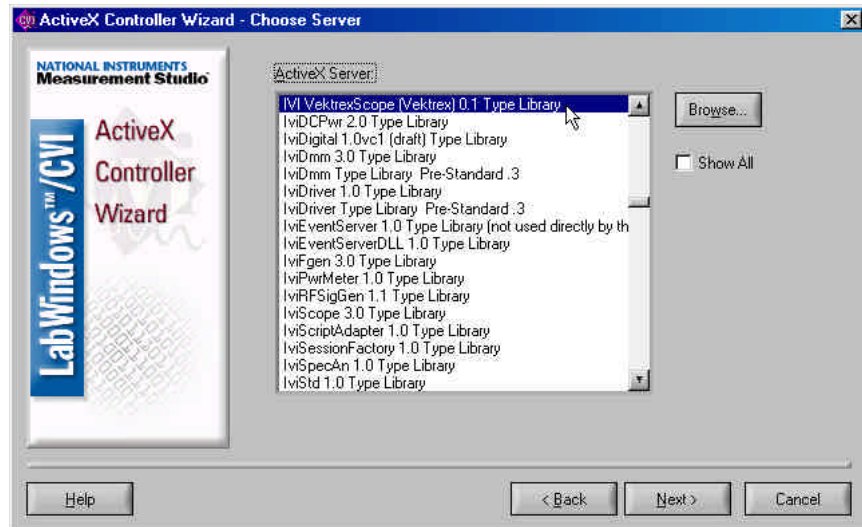
1. Start LabWindows/CVI and select *Create ActiveX Controller* from the *Tools* menu.



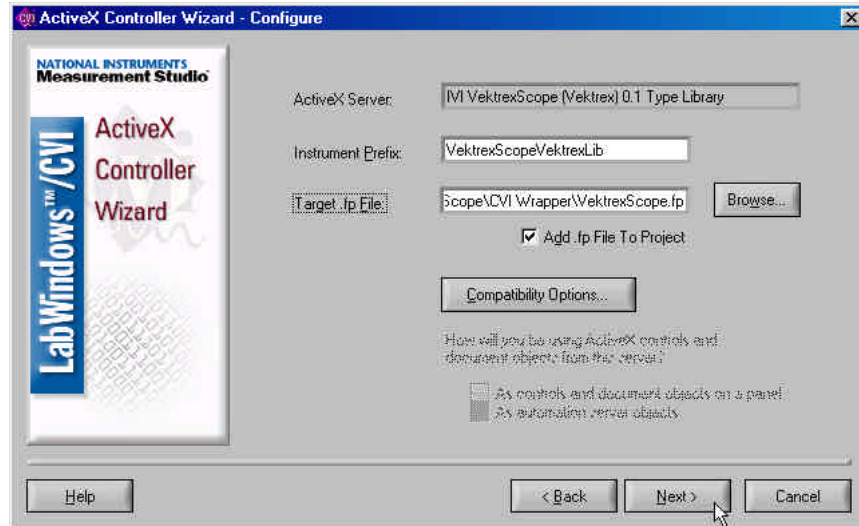
2. This starts the ActiveX Controller wizard (click on next...)



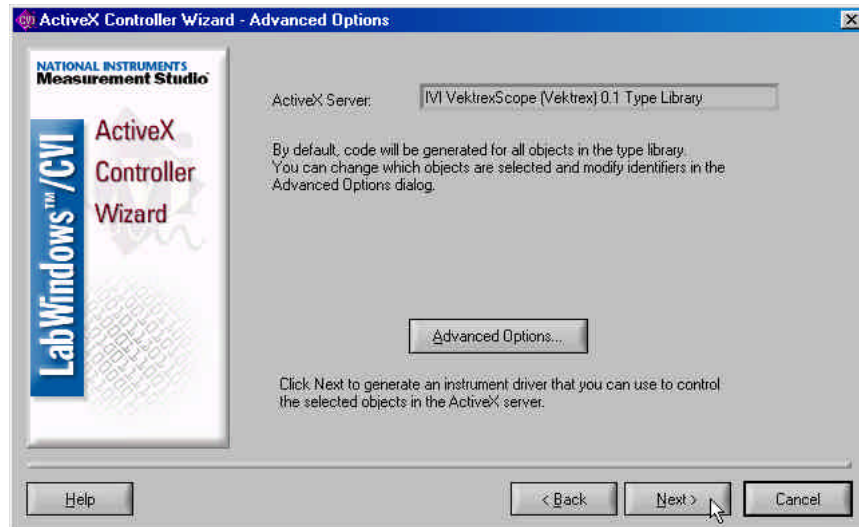
3. Select the IVI-COM driver from the list of registered components and click next....



4. Enter a path and name for the fp file that will be created by the wizard. The other settings can be left at the default....

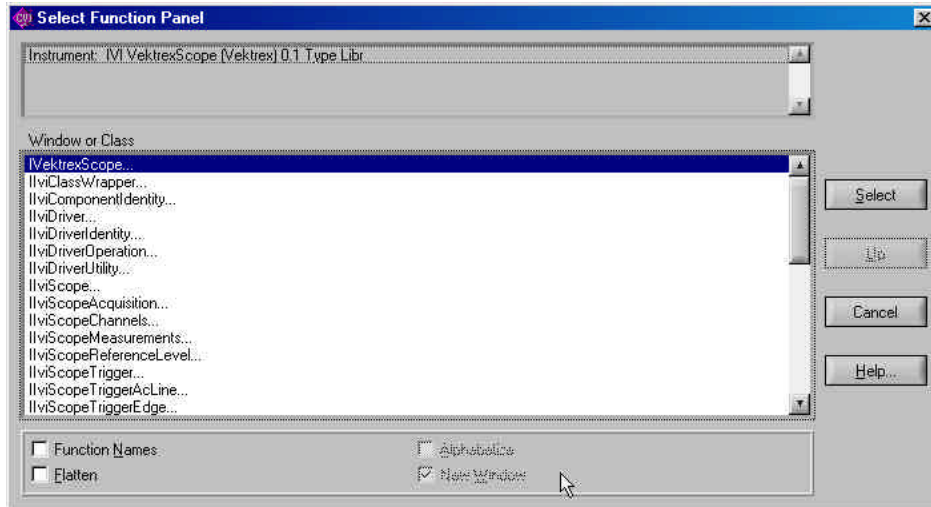


5. The “Advanced Options” step of the wizard allows you to choose which objects, properties and methods are included in the wrapper. The default is to generate code for everything. Since LabWindows/CVI does not need the interface reference properties that IVI-COM drivers contain for navigation around the driver, these could be removed from the wrapper in this step. In this application note we will not change anything in the advanced options step. Clicking next will generate the wrapper.

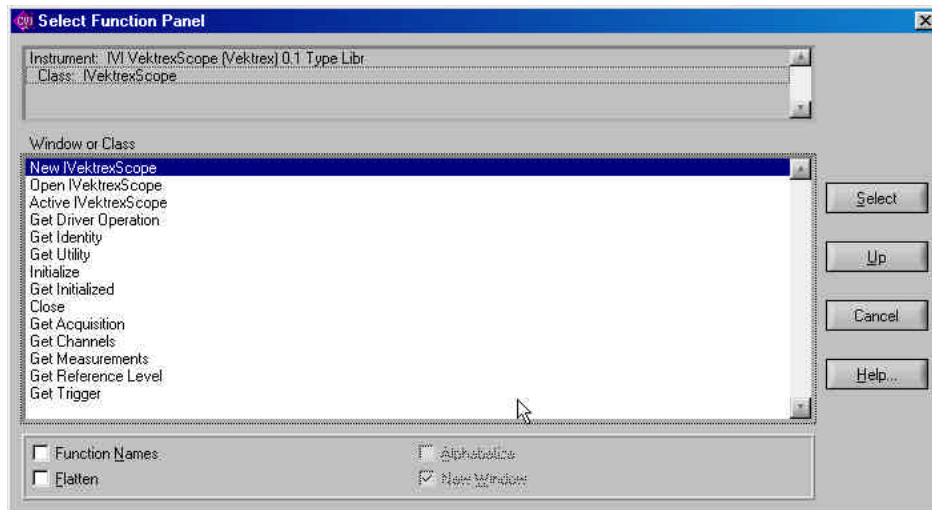


## Testing the Function Panels Interactively

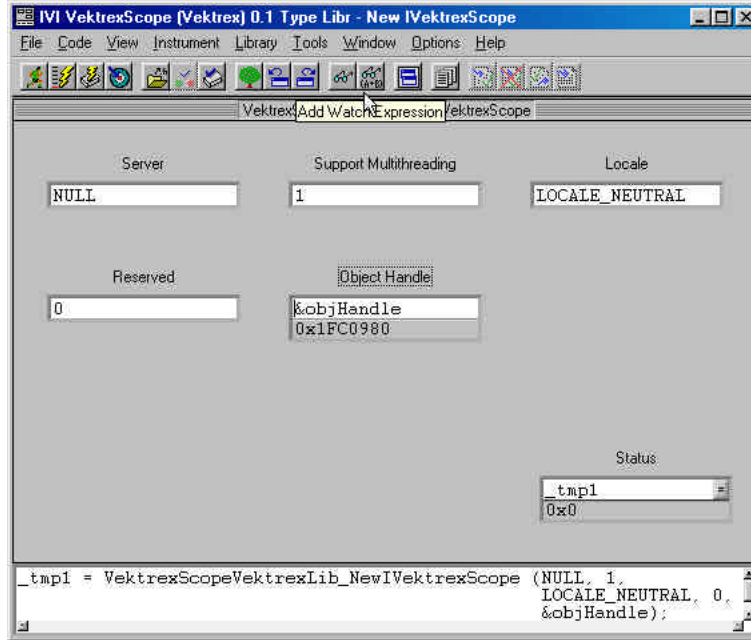
1. Double click the fp file in the project window to bring up the function selection panel.



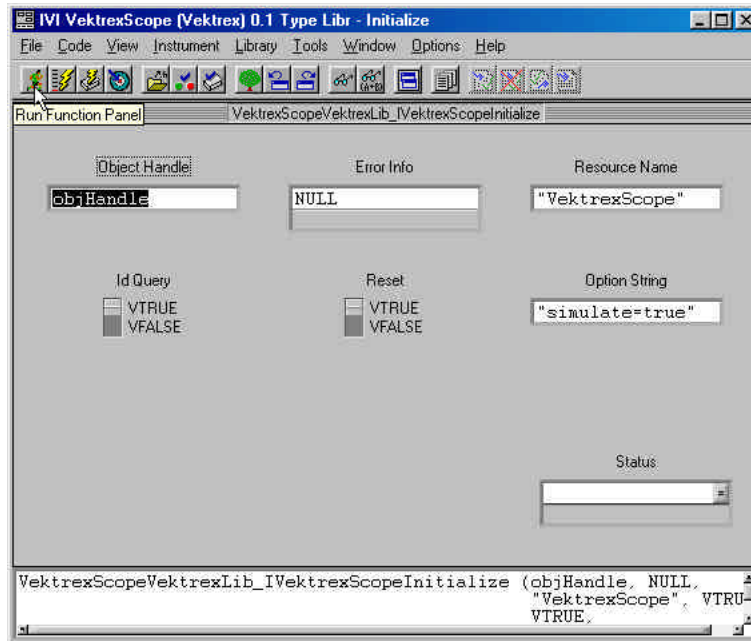
2. The first task is to create a new instance of the VektrexScope driver. Double click on IVektrexScope and then select New IVektrexScope.



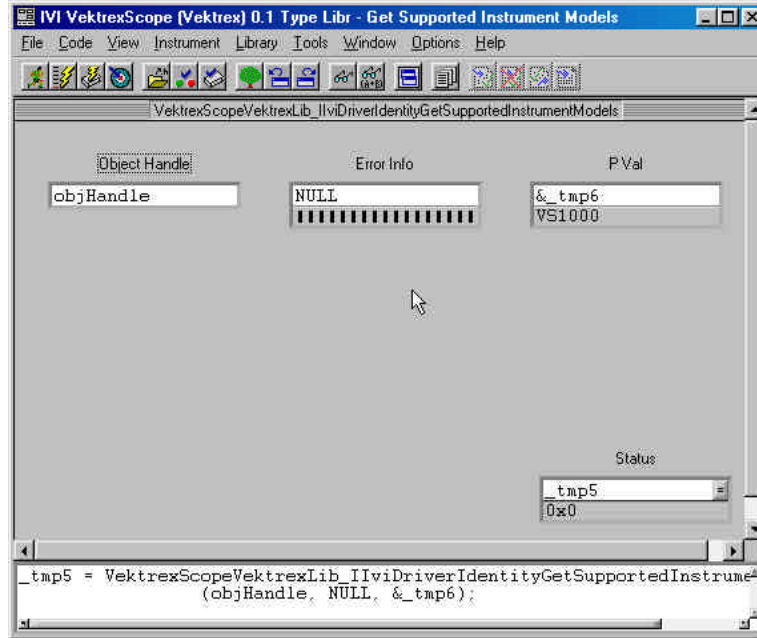
3. Declare a variable to hold the object handle that this function returns and run the panel. The handle will be used to run subsequent panels.



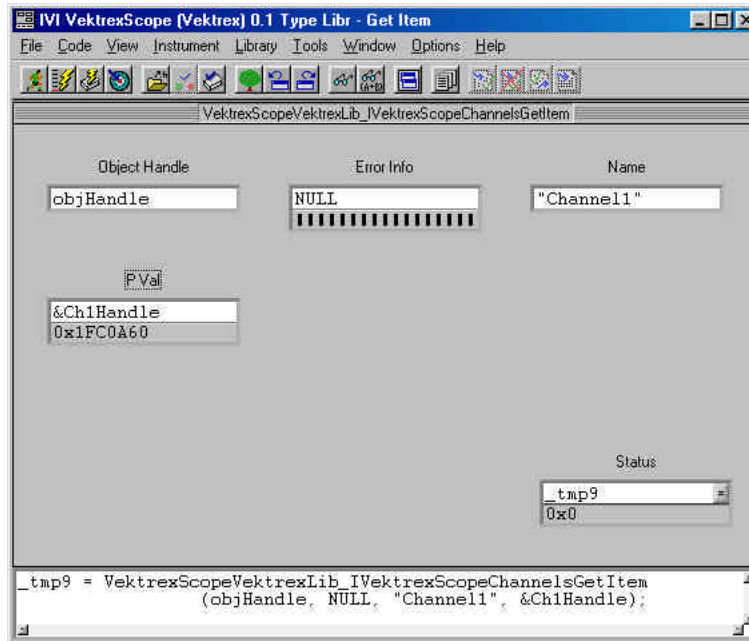
4. Next run the Initialize function. In this example we will use simulation.



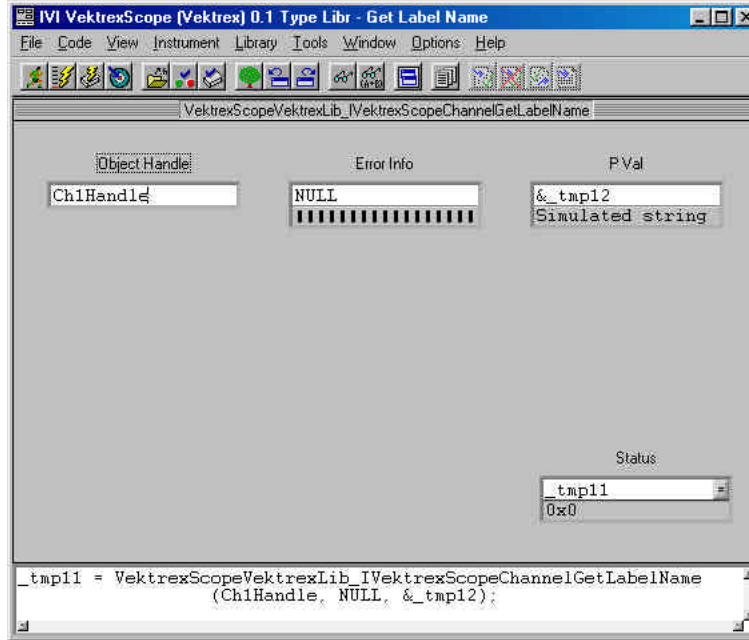
5. With the driver initialized we are free to execute any other function in the driver. Below is the result of running *GetSupportedInstrumentModels* from the *Identity* group.



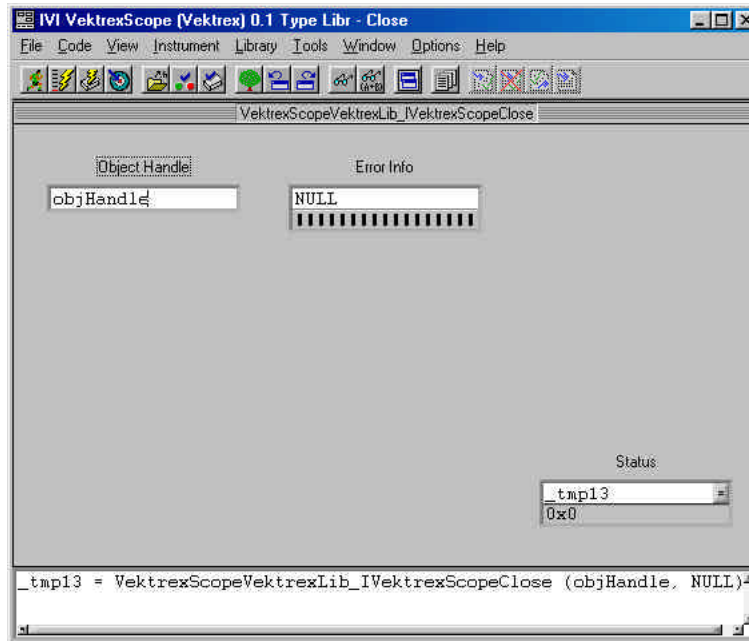
6. Accessing a repeated capability is a two-step process. First obtain a handle to the repeated capability using the *Item* function. Here we use *Item* from the *IVektrexScopeChannels* interface. Note that the handle returned (*Ch1Handle*) can be reused to call other functions in this interface.



7. The handle to the channel can be used to run functions within that interface – here *GetLabelName* is shown.

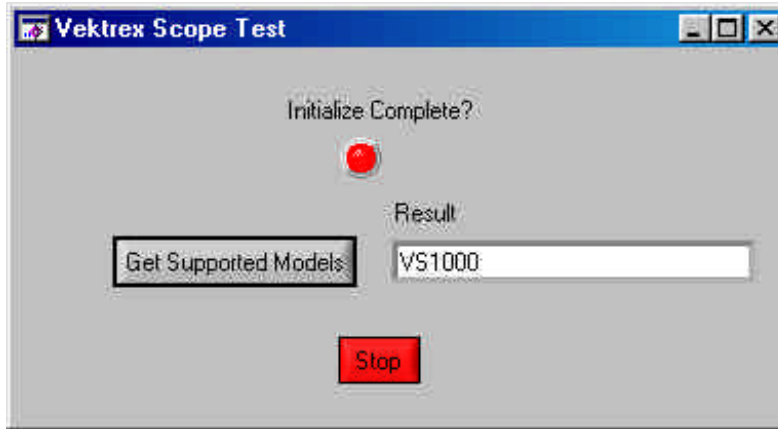


8. Don't forget to run the Close function at the end of testing.



## Using the Driver in a CVI Project

Inserting driver calls into a CVI project is achieved by filling in a function panel and then selecting *Insert Function Call* from the panel's *code* menu. Included with this application note is a simple application that shows some driver calls. This application has a user interface as shown below.



When the application is run, an instance of the driver is created and the Instrument is initialized. When this is complete the *Initialize Complete?* LED is lit. Pushing the *Get Supported Models* button causes the *StartAction* callback to run. This callback runs the driver's *GetSupportedInstruments* function, and displays the returned string in the *Result* box. If the *Stop* button is pressed the *StopButton* callback is run. This executes the driver's *Close* function and stops the application. The C code for this application is shown below.

```
#include "VektrexScope.h"
#include <cvirte.h>
#include <userint.h>
#include "TestVektrexScope.h"

static CAObjHandle ScopeHandle;
static int panelHandle;
static HRESULT hr;

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "TestVektrexScope.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);

    //Create instance of driver object
    hr = VektrexScopeVektrexLib_NewIVektrexScope (NULL, 1, LOCALE_NEUTRAL, 0,
                                                &ScopeHandle);

    if (hr == S_OK)
    {
        //initialize instrument
        hr = VektrexScopeVektrexLib_IVektrexScopeInitialize (ScopeHandle, NULL,

            "VektrexScope", VTRUE,

            VTRUE,

            "simulate=true");
    }

    if (hr == S_OK)
    {
```

```
        SetCtrlVal (panelHandle, PANEL_LED, 1);
        RunUserInterface ();
    }

    DiscardPanel (panelHandle);
    return 0;
}

int CVICALLBACK StartAction (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    char *SupportedModels;
    switch (event)
    {
        case EVENT_COMMIT:
            VektrexScopeVektrexLib_IIVIIdentityGetSupportedInstrumentModels
                (ScopeHandle, NULL, &SupportedModels);
            SetCtrlVal (panelHandle, PANEL_STRING, SupportedModels );

            break;
    }
    return 0;
}

int CVICALLBACK StopButton (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            VektrexScopeVektrexLib_IVektrexScopeClose (ScopeHandle, NULL);

            QuitUserInterface (0);
            break;
    }
    return 0;
}
```