Using the TekScope IVI-COM Driver from LabWindows/CVI

Introduction

LabWindows/CVI is a popular test-automation package from National Instruments. LabWindows uses the powerful C language to build Virtual Instruments. This document describes the step-by-step procedure for using the TekScope IVI-COM driver from LabWindows/CVI environment. LabWindows/CVI version 6.0 and higher support using Microsoft COM components.

In this simple exercise, you will learn how to import the TekScope IVI-COM driver from LabWindows environment using its *Create ActiveX Controller* feature. You will also learn how to use these wrappers to build a simple UI to connect to an oscilloscope and get current record length.

Requirements

The following software must be installed on your oscilloscope.

- TekVISA.
- IVI shared components.
- TekScope IVI-COM driver.
- LabWindows/CVI 6.0

Generating the wrapper FPs

Step 1: Start LabWindows

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

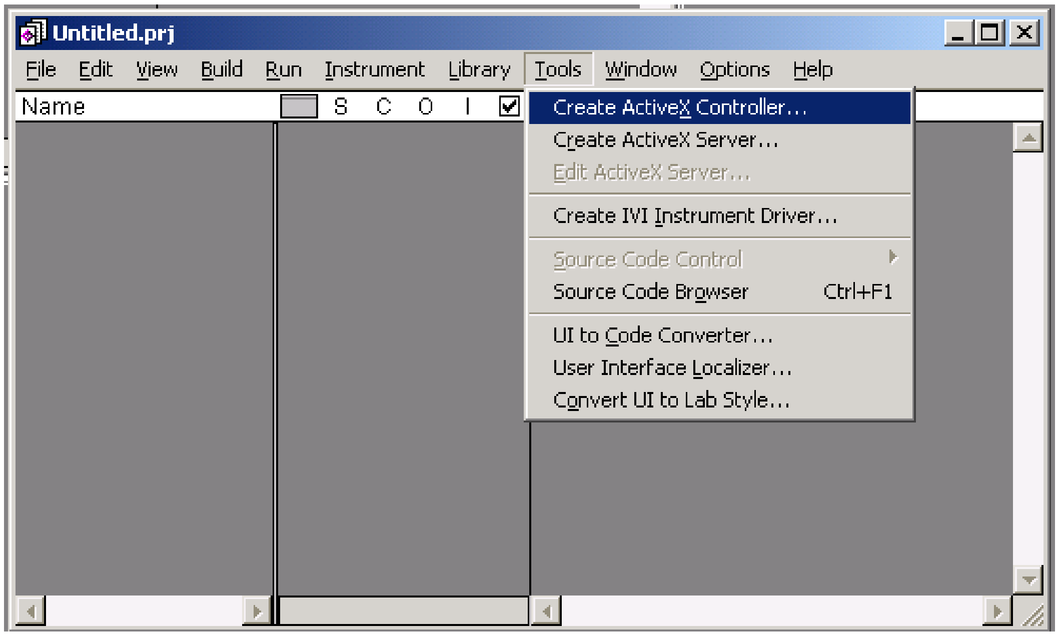
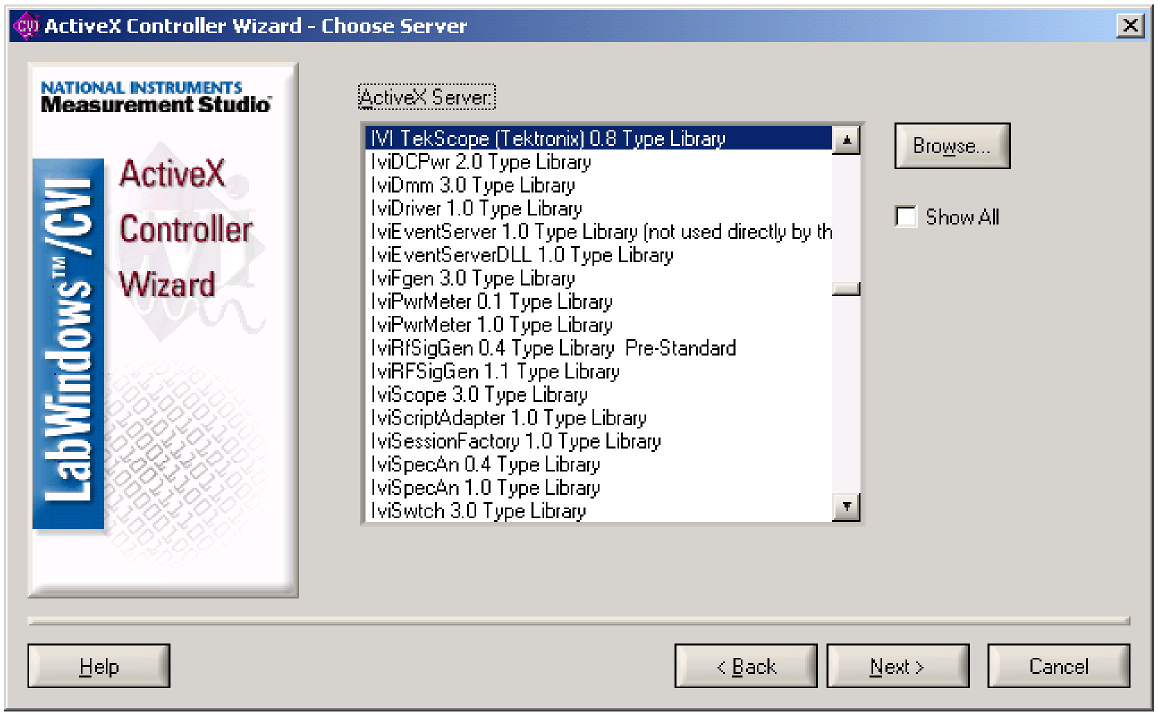

Untitled.prj
File Edit View Build Run Instrument Library Tools Window Options Help
Name S C O I
Create ActiveX Controller...
Create ActiveX Server...
Edit ActiveX Server...
Create IVI Instrument Driver...
Source Code Control
Source Code Browser Ctrl+F1
UI to Code Converter...
User Interface Localizer...
Convert UI to Lab Style...

Figure 1: Select *Create ActiveX Controller*

Step 2: Select the TekScope IVI-COM driver

Step 1 will bring up the *ActiveX Controller Wizard – Welcome* dialog. Click on the **Next** button in this dialog. The *ActiveX Controller Wizard – Choose Server* dialog box then comes up. This dialog lists all the ActiveX servers registered in the machine. From the list, select the TekScope IVI-COM driver, as shown in Figure 2, and click on **Next**.


ActiveX Controller Wizard - Choose Server
NATIONAL INSTRUMENTS Measurement Studio
LabWindows™/CVI
ActiveX Controller Wizard
Browse...
Show All
Help
< Back
Next >
Cancel

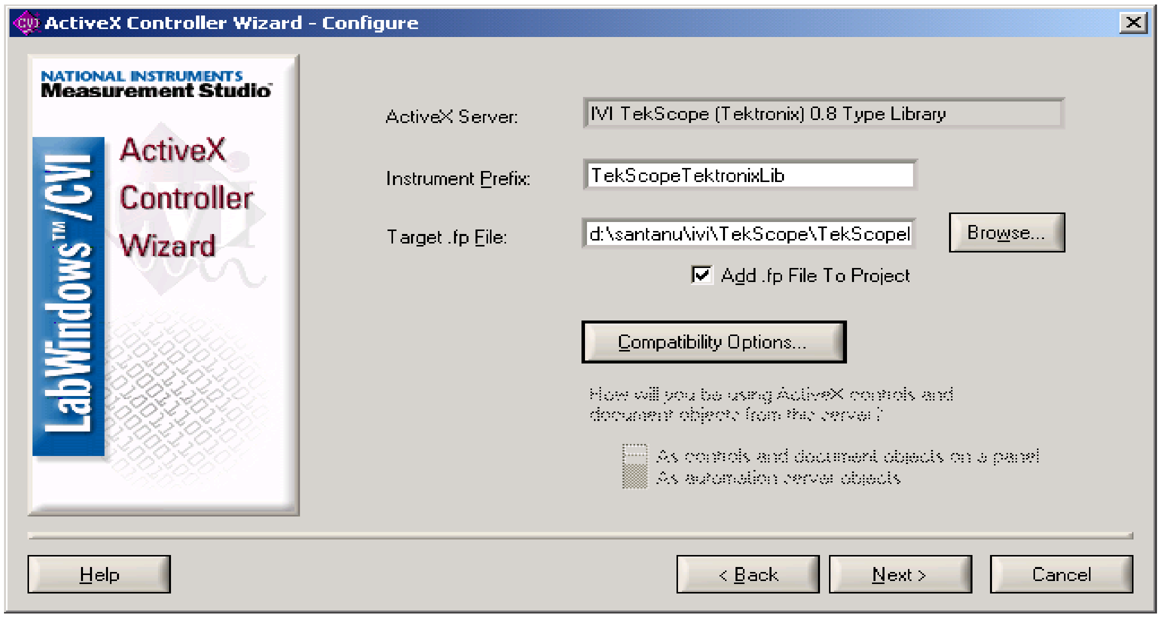
ActiveX Server:

IVI TekScope (Tektronix) 0.8 Type Library
IviDCPwr 2.0 Type Library
IviDmm 3.0 Type Library
IviDriver 1.0 Type Library
IviEventServer 1.0 Type Library (not used directly by th
IviEventServerDLL 1.0 Type Library
IviFgen 3.0 Type Library
IviPwrMeter 0.1 Type Library
IviPwrMeter 1.0 Type Library
IviRfSigGen 0.4 Type Library Pre-Standard
IviRFSigGen 1.1 Type Library
IviScope 3.0 Type Library
IviScriptAdapter 1.0 Type Library
IviSessionFactory 1.0 Type Library
IviSpecAn 0.4 Type Library
IviSpecAn 1.0 Type Library
IviSwtch 3.0 Type Library

Figure 2: The ActiveX Controller Wizard – Choose Server

Step 3: Create the file

In the *ActiveX Controller Wizard – Configure* dialog, specify a .fp file name (Ex: *TekScope.fp*) and path using the **Browse** button, as shown in Figure 3. This file will be created by the wizard, and all the wrappers for properties/methods of IVI driver are stored in this file. Click **Next** with other setting as defaults.


ActiveX Controller Wizard - Configure

ActiveX Server: IVI TekScope (Tektronix) 0.8 Type Library

Instrument Prefix: TekScopeTektronixLib

Target .fp File: d:\santanu\ivi\TekScope\TekScopeI

[✓] Add .fp File To Project

How will you be using ActiveX controls and document objects from the server?

As controls and document objects on a panel

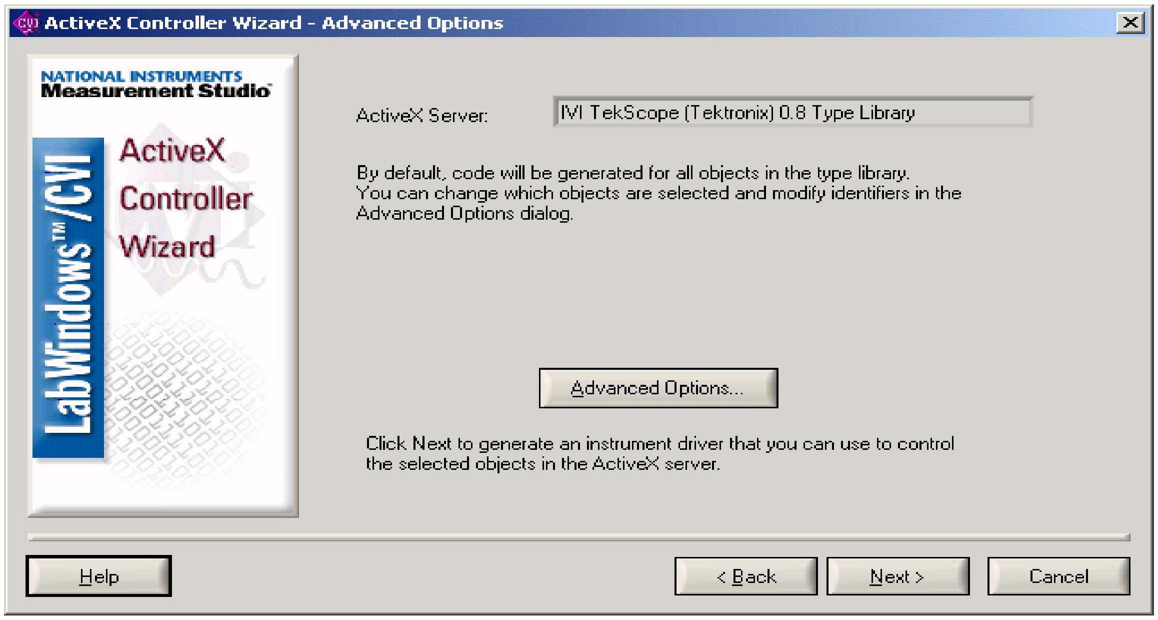
As automation server objects

Figure 3: The ActiveX Controller Wizard - Configure

Step 4: Advanced options

Step 3 brings up the *ActiveX Controller Wizard – Advanced Option* dialog, as shown in Figure 4. Use this ActiveX Controller Advanced Options dialog box to select the objects you want to include in the instrument driver and to change the names of the functions and properties that the wizard generates. You can also browse through all the interfaces, methods and properties of driver, as shown in Figure 5.

In this exercise, do not change anything in *Advanced Options* so that a wrapper is generated for the whole driver. Click **Next** to generate the wrapper FPs. Generating wrappers may take some time because the wizard also compiles all the wrappers during this process. After successful completion, you will get a message.

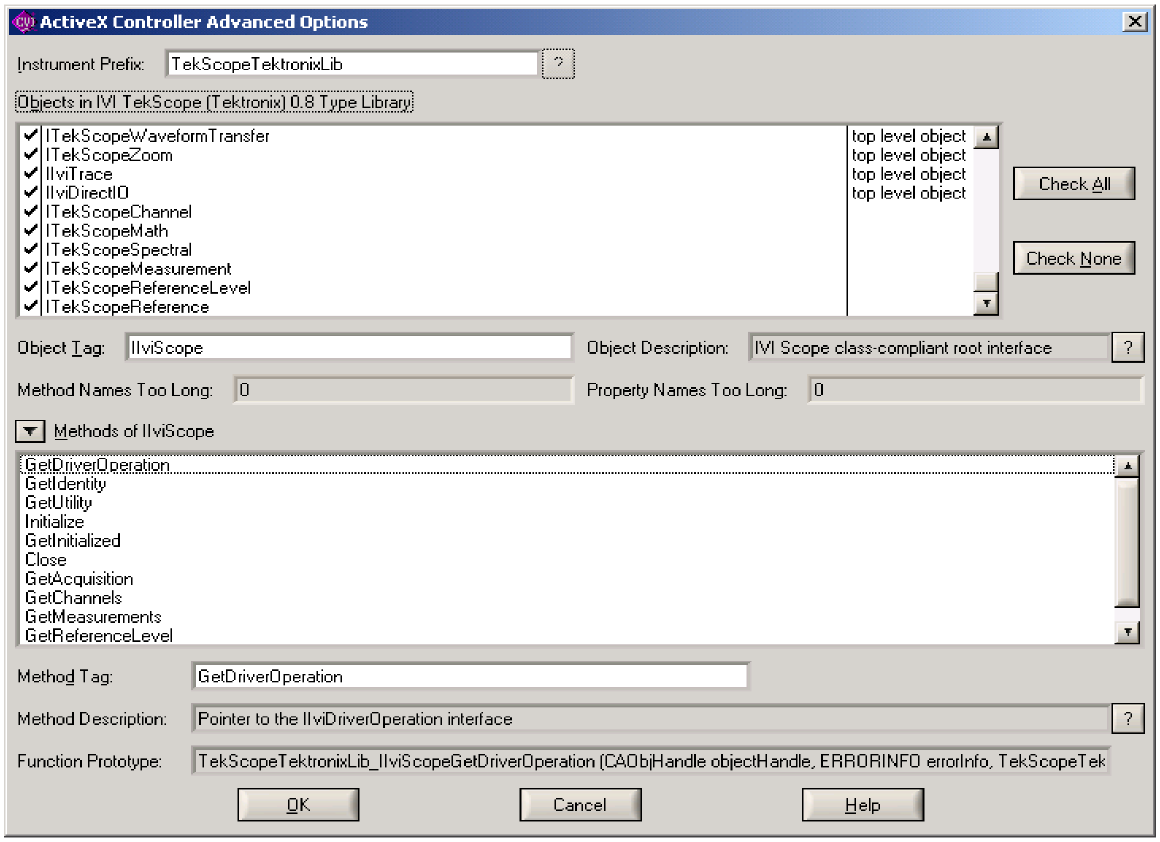

ActiveX Controller Wizard - Advanced Options

ActiveX Server: IVI TekScope (Tektronix) 0.8 Type Library

By default, code will be generated for all objects in the type library. You can change which objects are selected and modify identifiers in the Advanced Options dialog.

Click Next to generate an instrument driver that you can use to control the selected objects in the ActiveX server.

Figure 4: ActiveX Controller Wizard – Advanced Options


ActiveX Controller Advanced Options

Instrument Prefix: TekScopeTektronixLib

Objects in IVI TekScope (Tektronix) 0.8 Type Library

✓ ITekScopeWaveformTransfer	top level object
✓ ITekScopeZoom	top level object
✓ IIviTrace	top level object
✓ IIviDirectIO	top level object
✓ ITekScopeChannel	
✓ ITekScopeMath	
✓ ITekScopeSpectral	
✓ ITekScopeMeasurement	
✓ ITekScopeReferenceLevel	
✓ ITekScopeReference	

Object Tag: IIviScope
Object Description: IVI Scope class-compliant root interface
Method Names Too Long: 0
Property Names Too Long: 0

Methods of IIviScope

GetDriverOperation
GetIdentity
GetUtility
Initialize
GetInitialized
Close
GetAcquisition
GetChannels
GetMeasurements
GetReferenceLevel

Method Tag: GetDriverOperation
Method Description: Pointer to the IIviDriverOperation interface
Function Prototype: TekScopeTektronixLib_IIviScopeGetDriverOperation (CAObjHandle objectHandle, ERRORINFO errorInfo, TekScopeTek

Figure 5: All Driver Interfaces, Methods, and Properties

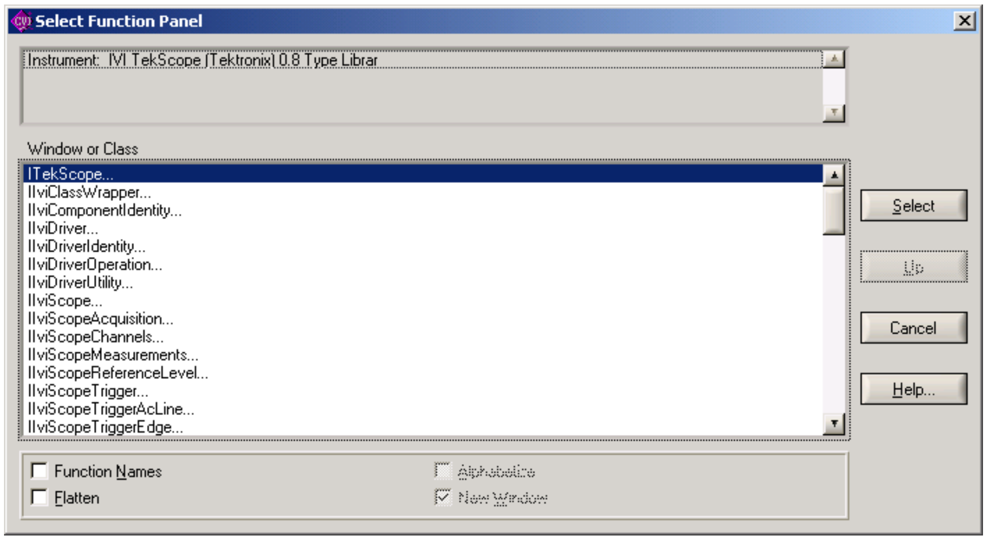
Step 5: Save

Once the wrapper generation is complete, *TekScope.fp* file is automatically added to LabWindows default *Untitled.prj* project. Save this project in the name of *CVISample.prj*.

Testing the Wrapper FPs

Step 1: Display the *Select Function Panel* window

Double click on the *TekScope.fp* file in the LabWindows environment to display the *Select Function Panel* window, as shown in Figure 6.



Instrument: IVI TekScope (Tektronix) 0.8 Type Librar

Window or Class

ITekScope...
IIviClassWrapper...
IIviComponentIdentity...
IIviDriver...
IIviDriverIdentity...
IIviDriverOperation...
IIviDriverUtility...
IIviScope...
IIviScopeAcquisition...
IIviScopeChannels...
IIviScopeMeasurements...
IIviScopeReferenceLevel...
IIviScopeTrigger...
IIviScopeTriggerAcLine...
IIviScopeTriggerEdge...

Figure 6: The Select Functions Panel window

Step 2: List all driver top level functions

Double click on the *ITekScope…* to list all top level functions of driver, as shown in Figure 7.

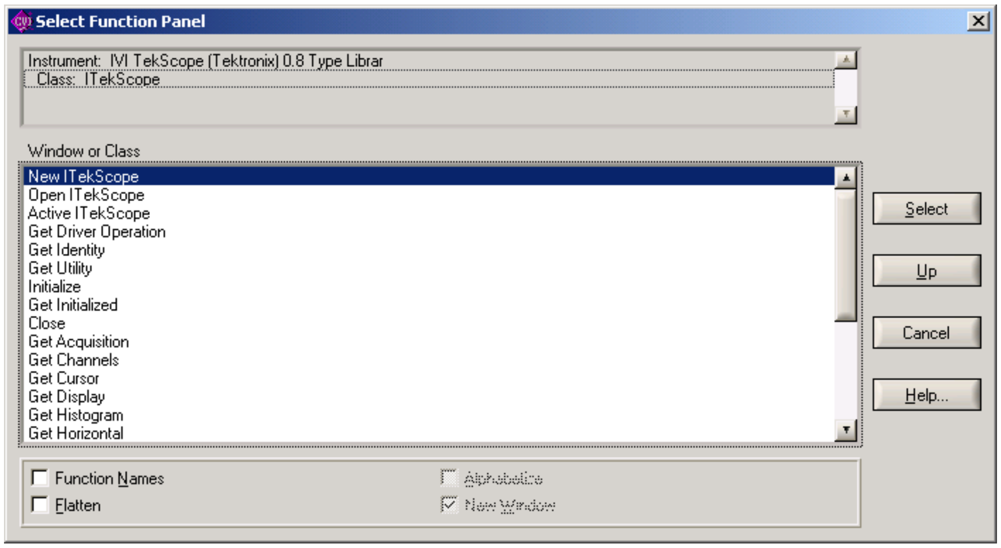
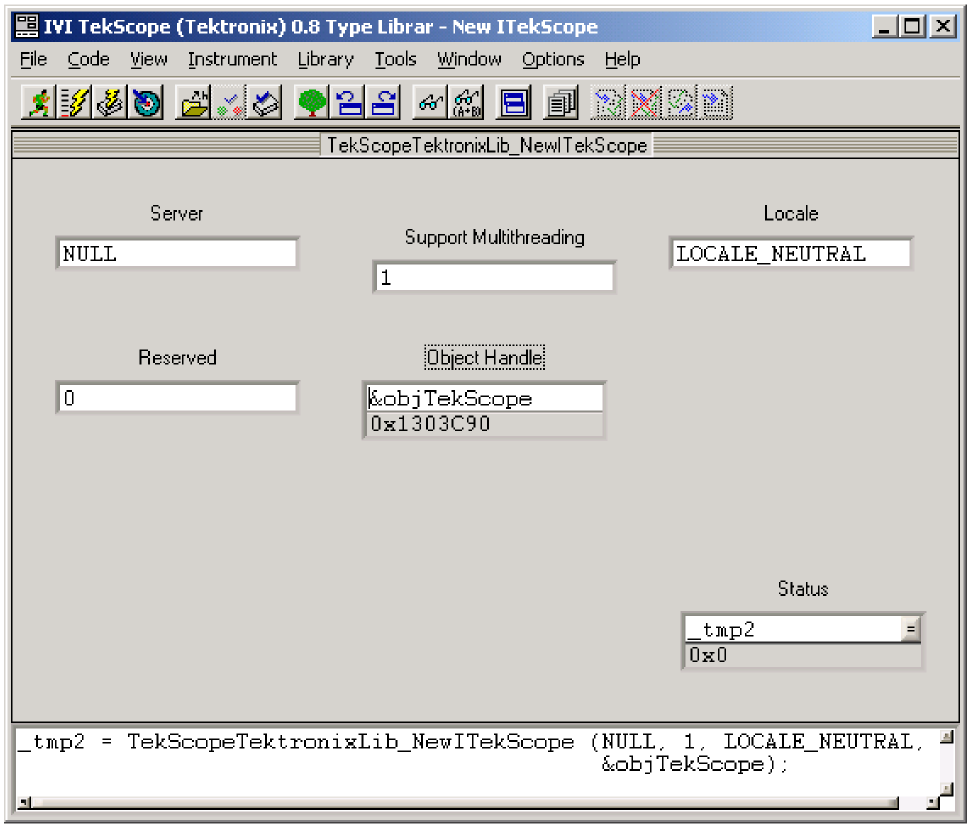


Figure 7: Top Level Functions

Step 3: Run the FP

Double click on the *New ITekScope* to open the FP, as shown in Figure 8. This function is used to create a new ITekScope object, and obtain a handle to the object. The obtained handle is used to call other functions of the driver. Declare a variable named *objTekScope* for the *Object Handle* control by first clicking on this control and then using **Code** > **Declare Variable…**. Run the FP.


IVI TekScope (Tektronix) 0.8 Type Librar - New ITekScope
File Code View Instrument Library Tools Window Options Help
TekScopeTektronixLib_NewITekScope

Server: NULL

Support Multithreading: 1

Locale: LOCALE_NEUTRAL

Reserved: 0

Object Handle: &objTekScope 0x1303C90

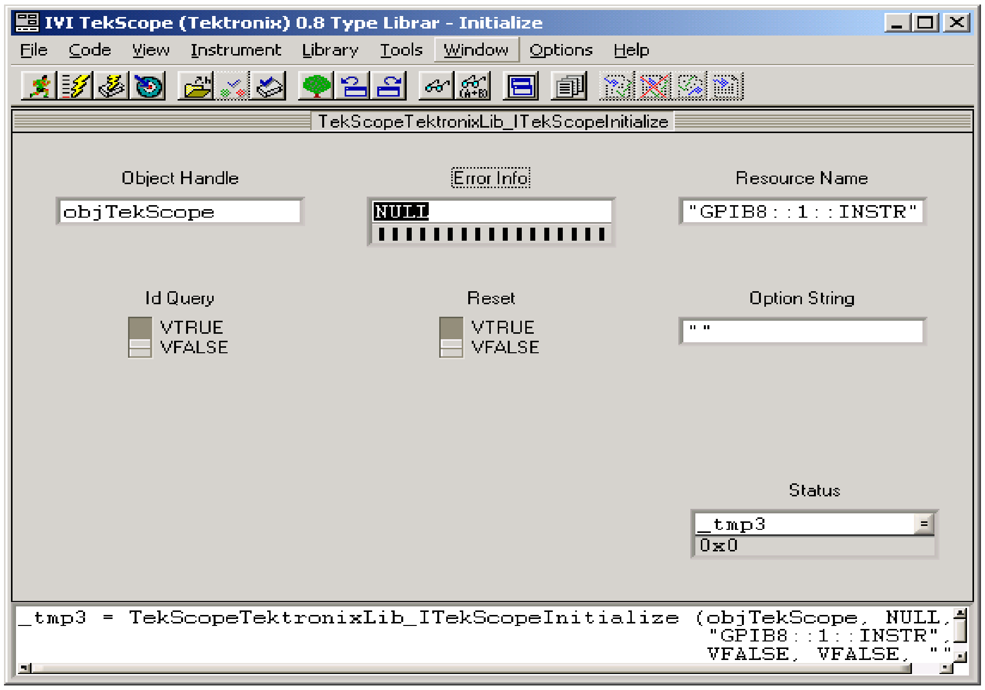
Status: _tmp2 0x0

```
_tmp2 = TekScopeTektronixLib_NewITekScope (NULL, 1, LOCALE_NEUTRAL,
                                           &objTekScope);
```

Figure 8: The FP Opened

Step 4: Run the FP

Open the Initialize function panel. In the function panel specify the *Object Handle* as *objTekScope* and *Resource Name* as VISA resource name. Here we have used *GPIB8::1::INSTR* as the resource name. Run the FP, as shown in Figure 9.


IVI TekScope (Tektronix) 0.8 Type Librar - Initialize
File Code View Instrument Library Tools Window Options Help
TekScopeTektronixLib_ITekScopeInitialize

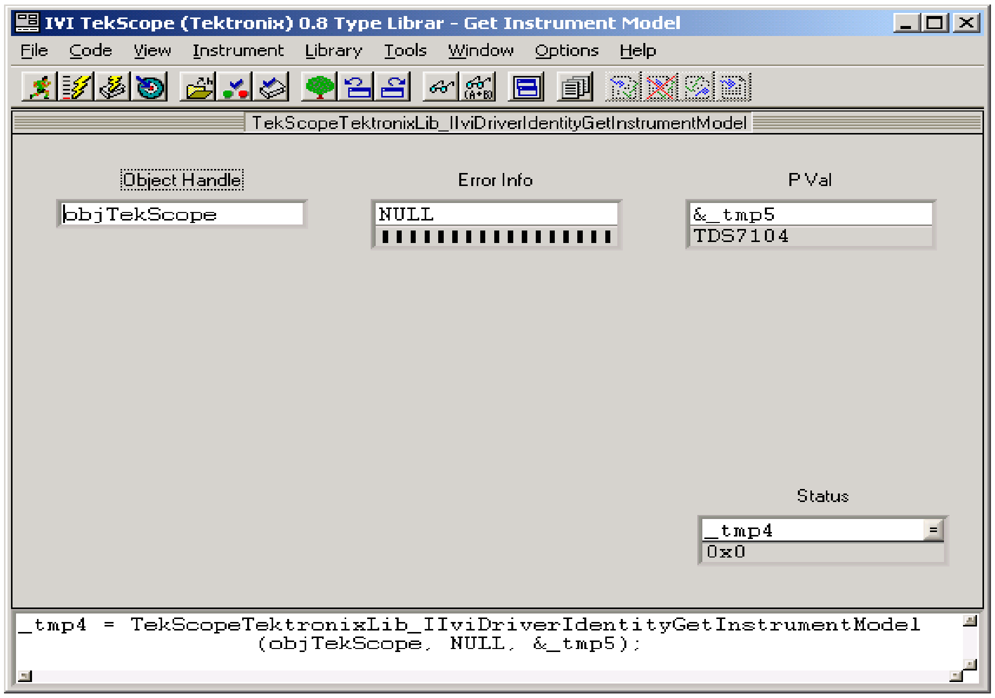
Object Handle: objTekScope
Error Info: NULL
Resource Name: "GPIB8::1::INSTR"
Id Query: VTRUE / VFALSE
Reset: VTRUE / VFALSE
Option String: ""
Status: _tmp3 0x0

```
_tmp3 = TekScopeTektronixLib_ITekScopeInitialize (objTekScope, NULL,
                                                  "GPIB8::1::INSTR",
                                                  VFALSE, VFALSE, "",
```

Figure 9: The FP Run

Step 5: Test driver functionality

To test the driver functionality, run a simple function Get *Instrument Model*. Open this function from the *IviDriverIdentity* class, specify the same object handle and run it. After running successfully the instrument model is returned. In the example shown in Figure 10, the model returned is *TDS7104*.


IVI TekScope (Tektronix) 0.8 Type Librar - Get Instrument Model
File Code View Instrument Library Tools Window Options Help
TekScopeTektronixLib_IIviDriverIdentityGetInstrumentModel

Object Handle: objTekScope

Error Info: NULL

P Val: &_tmp5 TDS7104

Status: _tmp4 0x0

```
_tmp4 = TekScopeTektronixLib_IIviDriverIdentityGetInstrumentModel
                 (objTekScope, NULL, &_tmp5);
```

Figure 10: The Model Returned is *TDS7104*

Step 6: Close the oscilloscope connection

Similarly, open the *Close* function panel and run it to terminate the connection to the oscilloscope.

Developing a Simple CVI Application

Step 1: Create a new UI

From File menu select **New** > **User Interface (*.uir)…** to create a new user interface. Add to command buttons to the UI and change the captions of the buttons to *Record Length?* and *Exit*, as shown in Figure 11. Save the file as *CVIClient.uir*.

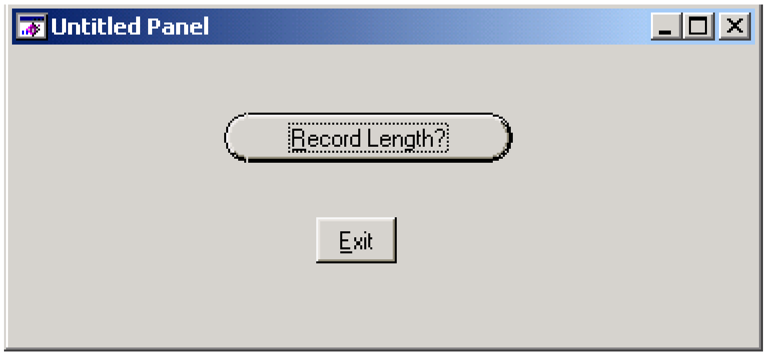


Figure 11: Changed Button Captions

Step 2: Create Skeleton Code

From the *CVIClient.uir* window select **Code** > **Generate** > **All Code…**. In the *Generate all Code* window, select the *Add To Current Project* option of the *Target files* drop down list and then press **OK** to generate a *cviclient.c* file. This C file will have skeleton code ready for main and command button callbacks. Double click on this file in the project window to explore the code.

Step 3: More Code

The code is given in the following table. The code in blue are added to the default codes (in black) generated by the LabWindows. You can copy these blue colored codes to your existing .C file.

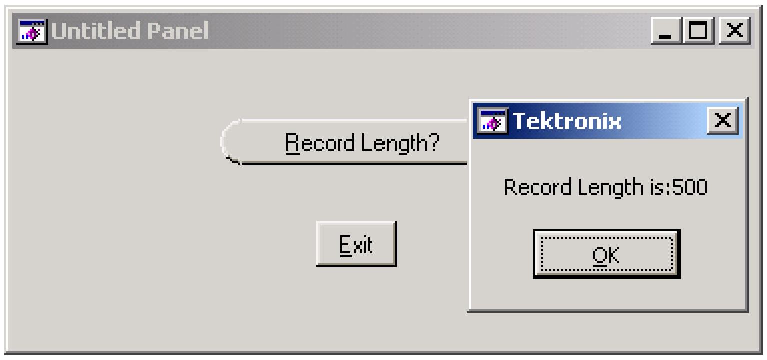
Note:

If you want to add additional driver functions to your C file, place your cursor in the C file where you want to add a function call and click. Open the function panel of that function. From the function panel window select **Code** > **Insert Function Call**.

```
#include "TekScope.h"

#include <ansi_c.h>

#include <cvirte.h>

#include <userint.h>

#include "CVIClient.h"


static int panelHandle;


//Declare the global variables

static CAObjHandle TekScopeHandle;       //for IVI-COM driver object

static HRESULT hr;  //for HRESULT value


int main (int argc, char *argv[])

{

        if (InitCVIRTE (0, argv, 0) == 0)

                return -1;       /* out of memory */

        if ((panelHandle = LoadPanel (0, "CVIClient.uir", PANEL)) < 0)

                return -1;

        DisplayPanel (panelHandle);


        // Create an instance of the TekScope IVI-COM driver

        hr  =  TekScopeTektronixLib_NewITekScope  (NULL,  1,  LOCALE_NEUTRAL,  0,
&TekScopeHandle);

        // if instance created successfully, initialize it

        if ( hr == S_OK)

        hr     =     TekScopeTektronixLib_ITekScopeInitialize     (TekScopeHandle,     NULL,
"GPIB8::1::INSTR",
        VFALSE, VFALSE,"");


        RunUserInterface ();

        DiscardPanel (panelHandle);

        return 0;

}


int CVICALLBACK GetRecordLength (int panel, int control, int event,

                void *callbackData, int eventData1, int eventData2)

{

        long nRecLen;

        char strMsg[256];


        switch (event)

                {

                case EVENT_COMMIT:

                        //Get the current record length of the oscilloscope

                        if ( hr == S_OK)

                        hr = TekScopeTektronixLib_ITekScopeHorizontalGetRecordLength
(TekScopeHandle, NULL,&nRecLen);
                        if ( hr == S_OK)

                        {
```

```
                                    sprintf(strMsg,"Record Length is:%d",nRecLen);

                                    MessagePopup ("Tektronix", strMsg);

                            }


                            break;

                    }

            return 0;

}


int CVICALLBACK OnExit (int panel, int control, int event,

                    void *callbackData, int eventData1, int eventData2)

{

            switch (event)
                    {

                    case EVENT_COMMIT:

                            if ( hr == S_OK)

                            TekScopeTektronixLib_ITekScopeClose (TekScopeHandle, NULL);

                            QuitUserInterface (0);

                            break;

                    }

            return 0;

}
```

Step 4: Run the application.



Record Length is:500

Conclusion

Similarly, you can use other driver functions from LabWindows/CVI.