ActiveX OptoMMP Technical Note

Introduction

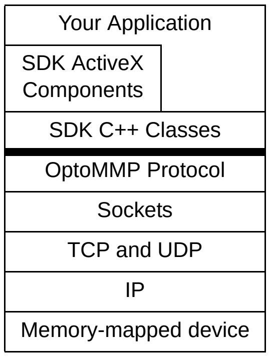
This technical note helps you use the ActiveX portion of the C++ OptoMMP Software Development Kit (SDK) for SNAP PAC (part number PAC-DEV-OPTOMMP-CPLUS, a free tookit available on our website at www.opto22.com.

This technical note assumes that you already have an understanding of ActiveX and C++ programming.

The SDK includes all the code for the C++ classes, ActiveX components, and examples. You are free to use this code at no charge in your own application, with or without modification. However, Opto 22 cannot support modification of the code.

The ActiveX components are essentially wrappers around the C++ classes, making the efficient C++ code available to many types of ActiveX clients.

The diagram at right shows how layers are related.



Overview of the ActiveX Components

You can use the ActiveX components from Windows programming environments that support ActiveX components, such as Borland Delphi, Microsoft Visual Basic®, Visual Basic for Applications, VBScript, and Visual C++.

Important ActiveX Client Issues

Before deciding to use these components in a particular language or tool, please keep the following in mind.

Some types of ActiveX clients are better suited for use with these components than others. The components are designed to be used in procedural languages, such as Visual Basic and C++. They are not recommended for use in other types of languages, such as graphical or dataflow languages (for exmaple, LabVIEW). Opto 22 recommends Visual Basic 6.0 and Visual Basic for Applications as the best environments for using these components.

Opto 22 is able to support the use of the ActiveX components only in the environments for which examples are provided: Access 2000, Word 2000, Visual Basic 6.0, Internet Explorer 6.0, Visual C++ 6.0, and Borland Delphi 5.

Also, please note these are ActiveX components, not controls. They do not have any user interface elements or design-time properties. They use methods and events but not properties.

Terminology

ActiveX is an umbrella term for many technologies based on COM (Component Object Model, a Microsoft technology allowing binary code reuse). Different programming environments use different terminology; in this chapter, we use terms as follows:

- An *interface* is a set of related functions (not a user interface).
- A *component* is an implementation of one or more interfaces.
- A *control* is a component that also has user interface elements.
- An *object* is an instance of a component.
- A *type library* is a description of a library's exposed classes, interfaces, methods, and properties. In this SDK, the type library is contained in the OptoSnapIoMemMapX.dll file.
- A *UUID* (Universally Unique Identifier, also called a GUID) is a a 128-bit value that uniquely identifies something.

ActiveX Names and IDs

NOTE: Different programming environments may require different information.

File Name: OptoSnapIoMemMapX.dll

Type Library UUID: 54D2FA40-E34F-11D2-9707-080009ABC65D

Type Library Name: OptoSnapIoMemMapXLib

MemMap ProgID: OptoSnapIoMemMapX.O22SnapIoMemMapX.1

MemMap Class UUID: 54D2FA50-E34F-11D2-9707-080009ABC65D

MemMap Class Name: OptoSnapIoMemMapX

MemMap Interface UUID: 54D2FA4F-E34F-11D2-9707-080009ABC65D

MemMap Interface Name: IO22SnapIoMemMapX

Streaming ProgID: OptoSnapIoMemMapX.OptoSnapIoStreamX.1

Streaming Class UUID: 54D2F150-E34F-11D2-9707-080009ABC65D

Streaming Class Name: OptoSnapIoStreamX

Streaming Interface UUID: 54D2F14F-E34F-11D2-9707-080009ABC65D

Streaming Interface Name: IO22SnapIoStreamX

Streaming Event Interface UUID: 54D2F148-E34F-11D2-9707-080009ABC65D

Streaming Event Interface Name: _IOptoSnapIoStreamXEvents

OptoSnapIoMemMapX General Instructions

Each programming environment has a different way of using ActiveX components. Also, environments use different and conflicting terminology. See the environment's documentation for more information on how to use ActiveX components. In general, however, you will need to do the following:

1. Add the component to the project, so the project knows about it.
   - In some environments, you must add a reference to the type library, class, or interface.
   - In others, you must import the type library, class, or interface into a native format. For example, if you are using the component in Visual C++, it will build a C++ wrapper class for accessing the component.
2. Create an instance of the component.
3. Open a connection to the Opto 22 memory-mapped device.
4. Use the methods of the component to communicate with the connected device and check for errors.
5. When done, disconnect from the device .
6. If needed, destroy the instance that was created in step 2.

OptoSnapIoStreamX General Instructions

Common procedures for using the OptoSnapIoStreamX ActiveX component in a program are:

1. Add the component to the project, so the project knows about it.
   - In some environments, you must add a reference to the type library, class, or interface.
   - In others, you must import the type library, class, or interface into a native format. For example, if you are using the component in Visual C++, it will build a C++ wrapper class for accessing the component.
2. Create an instance of the component.
3. Call OpenStreaming() to initialize the stream type, length, and port.
4. Call StartStreamListening() for each I/O unit.

   A new thread is created to listen for incoming UDP packets. Every time a stream packet from a registered device is received, the OnStreamEvent will be called.
5. Depending on the type set in step #3., call the function GetLastStreamStandardBlock(), GetLastStreamStandardBlockEx(), or GetLastStreamCustomBlockEx() every time OnStreamEvent is called.
6. To stop listening for a specified I/O unit, call StopStreamListening() at any time.
7. To add I/O units, call StartStreamListening() at any time.
8. When done, call CloseStreaming().
9. If needed, destroy the instance that was created in step 2.

Using Visual Basic 6.0 or Higher

When you add the component to a Visual Basic project, you must reference the component before using it. To do so, choose Project > References. In the list box, check OptoSnapIoMemMapX 2.0 Type Library. This action lets Visual Basic know that you intend to use the component from within the project.

Several sample programs are provided, including one (Demo Center.vbp) that demonstrates all aspects of the ActiveX component. The Visual Basic examples use the file ...\Examples\ActiveX\O22SIOMMXUtils.bas, which includes several useful general-purpose functions and error codes.

NOTE: If you're using a version of Visual Basic higher than 6.0, you may prefer to use the .NET OptoMMP Software Development Kit for SNAP PAC, also available for free download from our website at www.opto22.com.

Using VBA and Microsoft Office

Some Microsoft Office programs include a macro language, Visual Basic for Applications (VBA), which is a subset of Visual Basic. VBA can use the OptoSnapIoMemMapX component in the same way that Visual Basic does. Examples are included of using VBA in Word and Access.

Using VBScript

Visual Basic Scripting Edition, or VBScript, is a subset of the Visual Basic programming language. VBScript is an interpreted language used in World Wide Web browsers and other applications.

Microsoft FrontPage® or Microsoft Visual InterDev® can be used to write VBScript code that runs in an HTML page located on a PC on the same network as the Opto 22 memory-mapped device. An example is included that illustrates using VBScript in a Web page to access the ActiveX component.

If you are programming with VBScript, use the OptoSnapIoMemMap ActiveX component functions that begin with the letter V. These functions use only the variant data type, which is the only data type supported by VBScript.

Using Borland Delphi 5

Borland Delphi 5, a version of the Pascal programming language, may be used with the ActiveX components. To access them, the type library must be imported into a project. See the readme.txt file in the included Delphi example for more information.

Using the ActiveX Components without the SDK

The ActiveX component must be installed on a computer before it can be used. If you want to run the examples or other ActiveX files you create on a computer that doesn't have the SDK installed, you need to register the ActiveX DLL. To do so, follow these steps:

1. Copy the OptoSnapIoMemMapX.dll file to the new computer.
2. Register the OptoSnapIoMemMapX.dll file using the regsvr32.exe program.

   Regsvr32.exe can be found on most Windows systems. For instance, if you copied the ActiveX file to "c:\windows\system", you could register it by executing the command "regsvr32 c:\windows\system\OptoSnapIoMemMapX.dll" from a Command Prompt.

Building the ActiveX Component with Visual C++ 6.0

All Visual C++ 6.0 source code for the ActiveX component is included in the ...\Source\ ActiveX directory. This source code may be modified and built by using Visual C++ 6.0.

If the component's interface or functionality is changed, consider changing the library and interface's universally unique identifiers (UUID). These values can be found in the files OptoSnapIoMemMapX.idl and O22SnapIoMemMapX.rgs. Use the GUIDGEN.EXE program included with Visual C++ to generate new UUID values.