Application Note

8421-15950-08

EAN-08

WEIGH-TRONIX EAN-08

This paper describes what ActiveX technology is, what the benefits are to application program developers and how Weigh-Tronix intends to pursue it.

What is ActiveX ?

TITLE: WHAT IS ActiveX ?				
PART: 8421-15950-08	REV: A	ECO:	APPV'D: GJB	11-AUG-98

What is ActiveX ?

ActiveX is a technology that allows programmers to assemble reusable software components into application programs.

Although the *name* ActiveX is relative new, the underlying software technology has been around and evolving for many years. The interface standard that has emerged now allows software to be manufactured and sold as self-contained *software components*.

The Software Component Revolution

Ever since the early days of the personal computer, the software industry has been striving to achieve a level of software reuse like the hardware reuse enjoyed by the electronic industry. When a hardware developer sits down to design a complex logic board (a personal computer modem card for example), it is not necessary to do the design using all discrete components such as transistors. The integrated circuit industry packages transistors (and other components) into logic IC's which provide solutions to many standard and specialized logic problems. Further integration leads to single-chip solutions to many complex hardware tasks such as the serial communications controller device (UART) on a modem card. With this approach, many hardware development designs can be completed by breaking down a large project into smaller, (more manageable and well defined) tasks and then picking the appropriate hardware IC solutions from a catalog.

Software IC's (or software components) allow an application programmer to put together solutions in a similar way. In the software development industry there are generic problems that can use standard solutions provided by standard software components. There are also very unique problems, such as interfacing to scales, that can best be solved by using a specialized (or custom) software component. In either case, these components aid the developer by handling the simple common tasks that are likely to be needed, from one application to the next, as well as the complex task for which it is less expensive to buy the solution than it is to build one from scratch. Many specialized components vendors provide solutions that make it easy to justify buying especially when you consider the level of specialty experience, expertise and testing that is built in. Reusing tested, standardized code in this manner is called *component-based software development*.

Benefits of Component-based Software Development

Component-based software development has many advantages ove the traditional (monolithic) method of application development:

- It allows the developer to focus more on the overall project. The specific application solutions are built by integrating application-specific program code with the services provided by the software components. This speeds development and helps them to deliver the application in less time and at reduced cost.
- End-users of the application benefit by getting a higher quality product because it incorporates thoroughly tested components.

- Developers do not have to be experts in the area supported by the component. For example, a component to handle all of the RS-232 serial communications between a personal computer and a scale means that the developer does not need to understand the detail and nuances in the scales protocol specification just to get a weight value into the application. *In fact, they do not even need to understand the workings of serial communications in general!* Users need only understand the published interface document that accompanies the software component.
- A single component installed on a personal computer is all that is required for *any* number of applications that require the use of that component. *This is true even if two or more applications are running at the same time!*
- Each application program is smaller by the total size of software components used that are already on the computer because they can use the existing components. The software component only needs to be installed once on the computer. The code-size burden is not carried in each application program.
- Modular software applications built using software components makes maintenance much easier and extends the life of the application.

For example:

Imagine a new and important feature (or bug fix) has been made to a software component that is used to interface scales to a personal computer. Also, consider that there are multiple application programs on the same computer, all of which use that component. By updating just that **one** component, all applications that access it will immediately benefit from the changes that were made. The application programs themselves would not have to be modified. Contrast that to the alternative which would require the developer to modify each and every application with the desired changes and then rebuild the program.

What's an ActiveX Component ?

An *ActiveX component* is a generic term for a software component which provides useful *services* to an application program. It provides these services through an industry standard called **COM**. It's considered a generic term because there are a number of different types of ActiveX components.

ActiveX components must be used in a (32-bit) ActiveX compliant development environment such as Microsoft Visual Basic 5.0, Visual C++, Office 97, Visual FoxPro, Borland Delphi and others.

This means that the computer operating system (that the ActiveX components will ultimately run on), must also be 32-bit such as Windows 95.

Ok, ...so

what's COM ?

Simply stated, COM is a specification for how to build software components. Since any given application could have software components from different component vendors, it is important that they are all able to coexist, interoperate, install and work in a reliable defined manner with other software components. As long as the component is COM compliant, the developer can be assured that it will function with the development environment and with other components.
ActiveX components are considered to be a type of COM component. The important point in all of this is that ActiveX components must comply with strict guidelines and specifications in order to provide their main benefit which is code reusability.

Notes:
1) Because of COM, a vendor can develop the software component using a different programming language than what the application developer is using and everything will still work together!

2) Component software development using ActiveX technology should not be confused with object-oriented programming (OOP). OOP is a methodology used to build object-based software components, whereas ActiveX is a technology that allows you to combine those components and ensure that they work together.

Types of ActiveX Components

Although there are many forms of ActiveX components, the two that are of the most interest to us are the **ActiveX DLL** and the **ActiveX Control.**

What's an ActiveX DLL ?

An ActiveX **DLL** is a code component. Services provided in an ActiveX DLL package can be reused among different applications. Once the application programmer installs (ie *links*) the DLL into the application program, he has access to all of the services available in that DLL. Services include such things as:

1) The application can execute special routines (ie *methods*) in the DLL.
2) The application can read or write special attributes (ie *properties*) of data handled by the DLL.
3) The DLL can automatically notify the application that an important *event* has occured.

The application programmer must follow the published document that accompanies the DLL in order to be aware of all of the various services (ie methods, properties and events) that the DLL provides and how to use them.

What's an ActiveX Control ?

An ActiveX **Control**, in many ways, is like an ActiveX DLL. However, in addition to providing the services like the DLL does, it also provides a convenient *visual* user-interface. It's an ActiveX DLL wrapped in a user-interface. The word "user" here really refers to two groups of people. The first is the application program *developer* and the second is the *end-user* of the application program.

To the application program developer, the user interface provides many conveniences during the development process.

When the developer loads the ActiveX Control into the development environment, the control immediately becomes part of that environment. The control shows up *visually* as an **icon** along with all of the other tools available to the programmer. The developer can then *drag-and-drop* the control to place it where it is required in the application. This is called the *design-mode*.

Some controls will ultimately be accessed directly by the end-user during what is called *run-mode*. For example, a **text-box** control could be used to allow an end-user to key in a zip code. The developer can change the size of the text box (to whatever is desired), by simply clicking and dragging on special *sizing* handles located on the control. This is possible because ActiveX controls have built-in ability to draw themselves!

In addition, during design-mode, the control presents all of its properties to the developer in a convenient page format. The developer can set properties simply by clicking on desired values. In this way it is possible for the developer to gain a lot of functionality without yet having written a single line of code.

Note: Some ActiveX Controls do not have (or need) a visual user-interface for the end-user.

Where does Weigh-Tronix fit into all of this ?

We want to sell lots of scales and load cells to customers that will connect them to personal computers.

We provide additional value to our customers when we can offer ActiveX components that make their application program development easier. They are able to focus on their main application and leave the scale interface to us.

Currently, we can provide ActiveX Controls that handle various scale serial interface protocols including the Model 7010 family and the NCI Standard protocol. These *protocol handlers* provide simple solutions to non-trivial interface problems. An application developer can "drop" one of these controls into an ActiveX compliant "container", set a few properties and immediately have access to weight and status information. All of this without having to worry about the details of communications handshaking, error handling and retries, command parsing or protocol details. We do it for them.

The Future:

The next level of ActiveX controls will go beyond *just* handling this type of serial communications between the scale and a computer.

A future **ScaleX** ActiveX control will provide all of the weighing algorithms required to implement a working scale using just the raw counts from a transducer. The transducer may be a Quartz digital load cell or analog strain gauge-based converter board, or some other technology. These smart transducers will provide linear, compensated raw count values as their output. The ScaleX control will access those counts and convert them into usable net weight and status.

All other *scale-application* related routines (e.g. Zero, Auto-Zero_Tracking, Tare, Motion etc.) will be configured and provided by the ScaleX control. The ScaleX control (object) is not dependent on the specific type of transducer hardware. Instead, a *device-dependent* interface driver connects the ScaleX object with the transducer hardware. This is necessary because some transducers will use serial RS-232 interfaces, others could use RS-485 or even USB.

Applications developed in this manner allow the programmer to change operating parameters of the 'virtual' scale by setting property values in the control. For example, the range over which the scale may be zeroed can be changed to +/- 2% of full scale capacity by setting the properties **ZeroRangePlus** = 2 and **ZeroRangeMinus** = 2. This type of capability allows custom scale-based programs to be developed without the limitations of what built-in serial commands are supported by particular scales.

Also, the architecture allows any transducer type to be unplugged from the computer and replaced with a different type and the application will still function. For example; imagine a custom counting scale program developed on the PC using ScaleX. It may initially be connected to a low-cost 10 lb 7010 scale. Now, the count accuracy achievable may not be very impressive, but it would be possible to unplug the 7010 and plug in a QDT base. The same application would run, but the quality of counts available from QDT would be much better and therefore the count accuracy of the virtual counting scale would be enhanced greatly. This allows scaling up (and down) of hardware as required for applications without modifying the application software.

Weighing Products & Systems

Postal Scales

POS Scales

Dot Matrix Impact Printers

Thermal Graphic Label Printers

U-Mail® Desktop Mailing System

www.wt-nci.com

Weigh-Tronix Corp.
2320 Airport Blvd.
Santa Rosa, CA. 95403-1098

Tel: (707) 527-5555
Fax: (707) 527-5517