VBA TUTORIAL

smar FIRST IN FIELDBUS

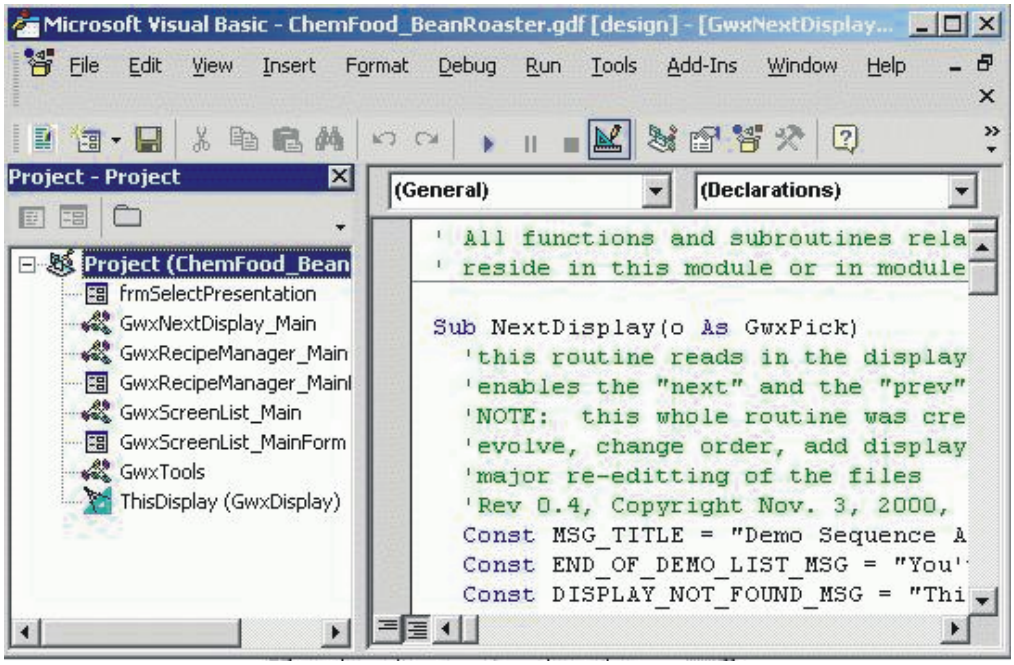
JUN / 04

PROCESS VIEW
VBA TUTORIAL
VERSION 7.1

FOUNDATION

INSTALLATION AND OPERATION MANUAL

PROCESS VIEW
VBA TUTORIAL



PVI EWTUTME

www.smar.com

Specifications and information are subject to change without notice.
Up-to-date address information is available on our website.

web: www.smar.com/contactus.asp

Index

Visual Basic for Applications

Introduction to Visual Basic for Applications

Visual Basic for Applications (VBA) is a Microsoft Visual Basic programming system Application Edition. It is an industry standard and a powerful programming environment. It is the fastest and easiest way to create and customize Microsoft Windows applications. ProcessView is shipped with Microsoft Visual Basic for Applications. VBA allows you to customize ProcessView to suit your specific requirements. It also offers high-level application programmability and features cross-platform support for ActiveX technology for the Microsoft Windows® operating systems. It is identical to VBA in Microsoft Office applications and other third-party products.

VBA allows you to:

- Create, debug, and run custom scripts or macros.
- Write Visual Basic code for events.
- Modify native objects.
- Connect ActiveX objects to each other and to native objects.

It allows both configuration and runtime operations.

This section documents only VBA topics related to ProcessView applications. For a complete VBA reference, please see the Microsoft Visual Basic for Applications Help documentation.

Visual Basic Concepts

It is important to understand some of the key aspects of Windows and Visual Basic before proceeding with development work with VBA in the ProcessView environment. This includes Windows, Events, Messages, etc., in the Microsoft Windows environment.

Windows

A **window** is a rectangular region with its own boundaries. Examples of windows include:

- An Explorer window in Windows 95, 98, 2000, or Windows NT.
- A document window in Microsoft Word.
- Dialog boxes, text boxes, message boxes, and command buttons.

A container is basically a window that contains one or more other windows, buttons, controls, etc. The Microsoft Windows Operating System manages all the windows by assigning a unique ID to each of them.

Events

Events are actions associated with a window. Events can occur through user actions, such as a mouse click or a key press, or even as a result of another window's actions. When an event occurs, a message is sent to the operating system, which broadcasts the message to other windows. Each window can take appropriate action based on its own instructions for dealing with that action. Handling of these events is called **Event Handling.** Examples of event handling include:

- Repainting of a window by itself when uncovered by another window.
- Closing, minimizing, or maximizing a window by clicking on the appropriate control.

Many of the standard events or messages are handled automatically by Visual Basic for Applications. Other events are exposed to you as event procedures, and you can write powerful code to deal with it, without having to deal with unnecessary details.

Object-Oriented Programming

Visual Basic is an **object-oriented programming language.** Unlike procedural languages, such as C or Basic, Visual Basic uses objects to create applications. Examples of objects include:

- Forms
- Controls
- ProcessView displays (e.g. a TrendWorX display)
- Databases

There are objects everywhere you look. You can create your own objects from a set of rules called classes. Objects and classes simplify your coding and increase code reuse.

Classes

Classes are sets of rules that define objects. Objects in Visual Basic are created from classes. Thus, an object is said to be "an instance of a class." The class defines an object's interfaces, default methods, and properties. Descriptions of classes are stored in **type libraries,** and can be viewed using **object browsers.**

Objects

Objects contain both their code and their data; in other words, objects are *encapsulated.* This makes them easier to maintain than traditional ways of writing code. Visual Basic objects have properties, methods, and events.

Properties and Methods

Properties are data that describe an object. **Methods** are things you can tell the object to do. **Events** are things the object does; you can write code to be executed when events occur.

Development Using Visual Basic for Applications

VBA uses an event-driven model approach for development. The execution of the code is driven by events. Visual Basic interprets your code as you write it. You can write code, compile it, and test it during development. This saves a lot of development time because you can run the application as you develop it rather than waiting to compile it later.

VBA Editor

The VBA Editor is an Integrated Development Environment (IDE) that is integrated into the ProcessView applications. It can be opened from the TrendWorX, GraphWorX, and AlarmWorX menu bars by selecting **Macros > Visual Basic Editor** from the **Tools** menu, or by pressing the shortcut keys **ALT+F11.** This opens the Visual Basic Editor, shown in the figure below, which enables you to create, edit, debug, and run Visual Basic code. The custom code is stored in modules, class modules, and forms.

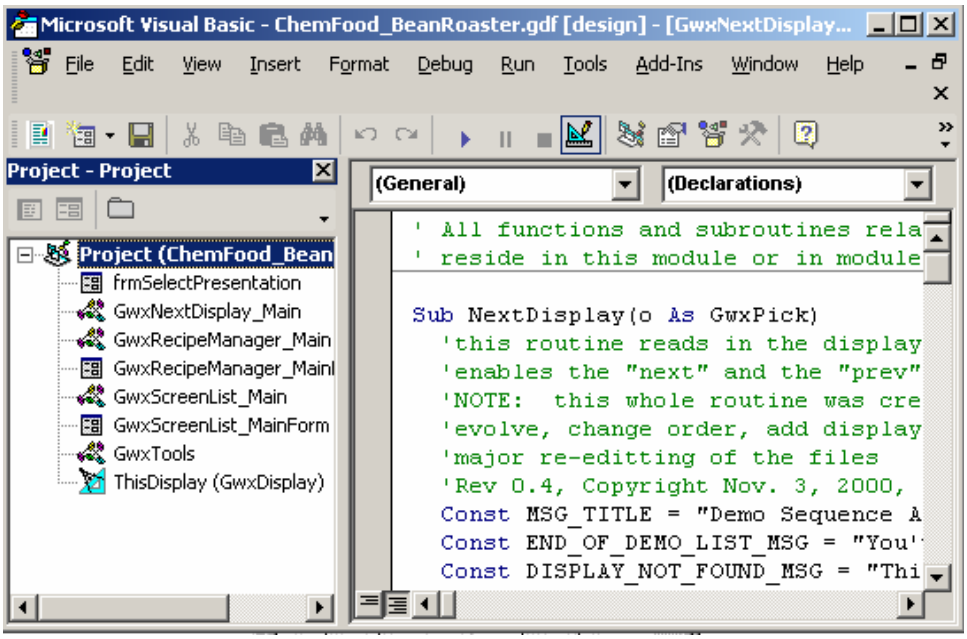


Figure 1. Visual Basic Editor

The VBA Editor supports project management. You can create projects using the editor. Projects can contain TrendWorX objects, GraphWorX or AlarmWorX objects, VB modules, forms, etc. Windows, such as the **Properties** and **Watch** windows, can be opened or closed from the **View** menu.

The VBA Editor consists of the following elements:

- **Menu bar**
- **Context menus**
- **Toolbars**
- **Toolbox**
- **Project Explorer Window**
- **Working With Projects**
- **Project file**
- **Properties window**
- **Object browser**
- **Form designer**
- **Code editor window**
- **Form layout window**
- **Immediate, locals, and watch windows**
- **Forms and controls**
- **ActiveX**
- **Modules**

Menu Bar

The **Menu** bar displays the commands you use to work with Visual Basic for Applications. It consists of the standard **File, Edit, View, Insert, Format, Tools, Window,** and **Help** menus, as well as some specific menus, including **Debug, Run,** and **Add-ins.**

Context Menu

The **Context** menu, or right-click menu, can be opened by right-clicking the object you are using. The **Context** menu contains shortcuts to frequently used commands.

Toolbars

The toolbars provide quick access to commands in the programming environment. By default, the **Standard** toolbar, shown below, is displayed when you start Visual Basic. You can choose to display the **Edit, Debug,** and **User Form** toolbars by selecting from the **Toolbars** submenu on the **View** menu.

Figure 2. Standard Toolbar

Toolbox

The **Toolbox,** which you can open by choosing **Toolbox** from the **View** menu, provides a set of tools that you use when designing to place controls on a form. In addition to the default toolbox layout, you can create your own custom layouts by right-clicking the Toolbox and selecting **Additional Controls** from the **Context** menu. You can and add available controls as needed.

Project Explorer Window

The **Project Explorer** window lists the forms and modules in your current project. To open the **Project Explorer** window, choose **Project Explorer** from the **View** menu, or press the shortcut keys **CTRL+R.** As you create, add, or remove editable files from a project, Visual Basic reflects your changes in the **Project Explorer** window, which contains a current list of the files in the project. The **Project Explorer** window shows some of the types of files you can include in a Visual Basic project.

Working With Projects

A **project** is the collection of files you use to build an application. You work with a project to manage all the different files that make up the application. Specifically, the VBA "project" is a collection of modules. All modules are stored along with objects to the same file (e.g. a .gdf file).

However, it is possible to export the modules to .bas files, the class modules to .cls files, and the forms to .frm files. A project consists of:

- One project file that keeps track of all the components (.vbp).
- One file for each form (.frm).
- One binary data file for each form containing data for properties of controls on the form (.frx). These files are not editable and are automatically generated for any .frm file that contains binary properties, such as **Picture** or **Icon.**
- Optionally, one file for each class module (.cls).
- Optionally, one file for each standard module (.bas).
- Optionally, one or more files containing ActiveX controls (.ocx).
- Optionally, a single resource file (.res).

Project File

The **project file** is simply a list of all the files and objects associated with the project, as well as information on the environment options that you set. This information is updated every time you save the project. All of the files and objects can be shared by other projects as well.

Properties Window

The **Properties** window, shown in the figure below, lists the property settings for the selected form or control. A **property** is a characteristic of an object, such as size, caption, or color. To open the **Properties** window, choose **Properties Window** from the **View** menu, or press the shortcut key **F4.**

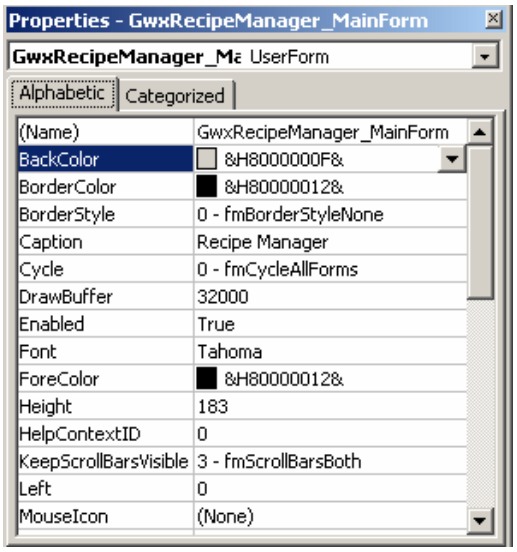


Figure 3. Properties Window

Object Browser

The **Object Browser** lists objects available for use in your project and provides you with a quick way to navigate through your code. To open the Object Browser, choose **Object Browser** from the **View** menu, or press the shortcut key **F2.** You can use the Object Browser to:

- Explore objects in Visual Basic and other applications.
- See what methods and properties are available for those objects.
- Paste code procedures into your application.

Form Designer

The **Form Designer** serves as a window that you customize to design the interface of your application. You can add controls, graphics, and pictures to a form to create the look you want. Each form in your application has its own Form Designer window.

Code Editor Window

The **Code** window serves as an editor for entering application code. A separate **Code** window is created for each form or code module in your application.

Form Layout Window

When you insert a new VBA form, the **Form Layout** window will appear. The **Form Layout** window allows you to position the forms in your application using a small graphical representation of the screen. All tools for inserting and positioning controls are available within the visible toolbars or menus.

Immediate, Locals, and Watch Windows

The **Immediate, Locals,** and **Watch** windows are provided for use in debugging your application. These windows are only available when you are running your application within the editor.

Forms and Controls

Forms are user interfaces, the visual part of the application with which you interact. Forms and controls are the basic building blocks used to create the interface; they are the objects that you will work with.

Forms are objects that expose properties that define their appearance, methods (which define their behavior) and events (which define their interaction with you). By setting the properties of the form and writing Visual Basic code to respond to its events, you customize the object to meet your requirements.

Controls are objects that are contained within form objects. Each type of control has its own set of properties, methods, and events that make it suitable for a particular purpose. Examples of controls are fields for entering or displaying text. Controls can also be used to access other applications and process data as if the remote application was part of your code.

ActiveX

ActiveX is a set of integration technologies that enables software components to interact in a networked environment using any language. ActiveX is based on Microsoft's Object Linking and Embedding (OLE) and the Component Object Model (COM).

ActiveX Control

ActiveX is a type of control that is an extension of the Visual Basic Toolbox. You use ActiveX controls just as you would any of the standard built-in controls, such as the **CheckBox** control. When you add an ActiveX control to a program, it becomes part of the development and runtime environments and provides your application with new functionality.

How Is ActiveX Used With ProcessView?

An **ActiveX Control** is basically used to embed objects from other applications into displays. Applications supporting ActiveX include GraphWorX, AlarmWorX, TrendWorX, and other Windows applications.

Modules

Code in Visual Basic for Applications is stored in modules. There are three different kinds of modules:

- **Standard modules**
- **Form modules**
- **Class modules**

Again, by default the modules of VBA for ProcessView are stored in .t32, .gdf, and .a32 files. They can be explicitly exported to files and imported back when needed.

Standard Modules

Usually the code associated with a form resides in that form module. If you have many forms or other modules that could use a common code, you can create a separate module containing a procedure that implements the common code. This separate module should be a **standard module**.

Each standard module can contain declarations, such as type and variable, and procedures, such as Function (Functions) or Sub (Sub routines). The standard module file has a .bas extension.

Form Modules

Form modules are the foundation of most Visual Basic applications. They can contain procedures that handle events, general procedures, and form-level declarations of variables, constants, types, and external procedures. The code that you write in a form module is specific to the particular application to which the form belongs. It might also reference other forms or objects within that application. Form module files have a .frm extension.

Class Modules

Class modules are the foundation of object-oriented programming in Visual Basic. You can write code in class modules to create new objects. These new objects can include your own customized properties and methods. Actually, forms are just class modules that can have controls placed on them and can display form windows. Class modules have a .cls file extension.

ActiveX Modules

The various ActiveX modules you can have are **ActiveX Documents, ActiveX Designers,** and **User Controls.** From the standpoint of writing code, these modules should be considered the same as form modules.