

An Intro to Visual Basic 5.0 Programming

Visual Basic 5.0 is a graphics (symbols or icons) based programming language based on BASIC (Beginners All-Purpose Symbolic Instruction Code). It is the computer language of choice among those of us who design software to interface to the real world. It provides an easy operator interface because the operator uses a mouse or keyboard to select 'buttons' or other 'controls' to cause things to happen or see things happen with easy to understand 'layouts' and graphics.

Visual Basic actually comes in three versions. The Learning edition for beginners, the Professional edition used for our projects, and the Enterprise edition for distributed applications in a team setting. The Professional edition has ActiveX components plus many other features beyond the Learning edition. One of the ActiveX components is MSComm and that's how we communicate through the ports of your computer to the real world. Let's get started with this most important component of this language from an interface point of view.

MSComm

Microsoft figured out a long time ago that since computers are linked to the real world, there had to be an easy way to interface the program registers in computers to outside devices such as the serial and parallel ports. After Visual Basic is started, you install and register an ActiveX component as an extension to the Visual Basic Toolbox by selecting it from a list and checking a box to activate it. Then it's selected icon appears in the toolbox window and is ready for use. In other words, it's a special tool that is added to your toolbox that takes care of a special need. Like those special ignition wrenches for loosening your distributor to change your timing or a spline screwdriver to remove screws for the headlights on your car. After fumbling around for a while, you figure out that you need something special to accomplish the job you set out to do. Now, let's take an in-depth look at what MSComm is all about.

There's a total of 32 separate properties that are associated with MSComm. Now before the number overwhelms you, let's start with the five more important ones. I'll define them and then we'll take a closer look at the actual format used in each one and how they work together to read from and write to the input and output registers of the serial port.

CommPort	Sets and returns the comm port #
Settings	Sets and returns baud rate, parity, data bits, and stop bits as a string
PortOpen	Sets and returns the comm port state-opens and closes the port
Input	Returns and removes characters from the receive buffer
Output	Writes a string of characters to the transmit buffer.

Now I'm sure you can see how much you can do with just these five registers. The convention for using these commands is formatted as:

Object.Command = (Value) where:

Object = MSComm (such as 1, 2, etc.)

Command = name of property (such as Commport)

Value = (such as True, False, a number or a string)

As an example, to define a link to the serial port for Port 1 we would use:

```
MSComm1.CommPort = 1
```

And if you wanted to use Port 2 then you would use:

```
MSComm1.Commport = 2
```

And if you wanted to use both ports for some reason you would use:

```
MSComm1.CommPort = 1
```

```
MSComm2.CommPort = 2
```

So far, so good. Now for the settings property. It formats like this:

```
MSComm1.Settings = "9600, N, 8, 1"
```

Now, we've got Port 1 defined as MSComm1 at 9600 baud, no parity, 8 bits, 1 stop bit. Now all we need is to open the port like this:

```
MSComm1.PortOpen = True
```

Now we can receive data from the receive buffers. Other commands decide how big the buffer is, how many characters must be received to trigger an event for the computer to read the buffer. The same thing applies to the transmit buffer to determine it's size and how many characters are sent before they are actually transmitted. When you're through, you issue the command to close the port as:

```
MSComm1.PortOpen = False
```

I hope that we have given you a clear and simple overview about how you can begin communication with the serial port of your computer. Obviously, there is much more in-depth information to follow as we continue in this series of articles. Next time, we'll explore in depth all the registers and how to control the handshaking lines as well as read these signals to communicate with the outside world. See you next time.