

Exploring the Registers of MSComm

As outlined in the previous article, MSComm is an ActiveX component of Visual Basic 5.0 which allows us to communicate thru the serial port to the real world. We outlined the 5 most important registers and covered some basic info on how to set up the parameters for those registers. We will continue with a more in-depth look at the remaining registers and how you can put them to use in your own programs.

As was mentioned on the website, we are developing a series of projects using VB5 with our own RS-485 network I/O modules. As the series progresses, we will release VB5 standalone executable programs to use with these modules. We'll start with an upcoming article which will show you how to build a complete RS-232/RS-485 interface with \$ 15.00 !

MSComm

There are 32 properties of MSComm; we covered 5 of them in the previous article. Once again, 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 break)

Value = (true, false, integer, or string)

In alphabetical order the remainder are:

Object.**Break** = (**true or false**) causes the character transmission to pause when true and continues when false.

Object.**CDHolding** = (**true or false**) monitors the state of the Carrier Detect line (CD-pin 8 of DB-25) as true when the input is high and false when the input is low.

Object.**CommEvent** = (**integer value**) holds the most recent communication error or event is logged. This register contains a 4-digit 10xx number for 11 communication errors and a single digit numbers for 7 separate communication events.

Object.**CommID** = (**integer value**) returns a handle that identifies the communications device.

Object.**CTSHolding** = (**true or false**) monitors the state of the Clear To Send (CTS-pin 5 on DB-25) line as true when the input is high and false when the input is low.

Object.**DSR Holding** = (**true or false**) monitors the state of the Data Set Ready (DSR-pin 6 on DB-25) line as true when the input is high and false when the input is low.

Object.**DTR Enable** = (**true or false**) controls the state of the Data Terminal Ready (DTR-pin 20 of DB-25) as high when the value is true and low when the value is false.

Object.**EOF Enable** = (**true or false**) activates the OnComm event when an EOF character is received if true or ignores EOF characters when false.

Object.**Handshaking** = (**value**) where the value is 0-3 representing comNone (no handshaking), comXOnXOff (XON/XOFF handshaking), comRTS (RTS/CTS handshaking), and commRTSXOnOff (both types of handshaking) respectively.

Object.**InBufferCount** = (**value**) returns the number of characters in the receive buffer. You can clear the receive buffer by setting the property = 0.

Object.**InBufferSize** = (**value**) sets and returns the size of the receive buffer in bytes. The default size is 1024 bytes.

Object.**Index** = (**expression**) will return or set the number that uniquely identifies an object in a collection (ActiveX component).

Object[(number)].**Index** will return or set the number that uniquely identifies a control in a control array.

Object.**Input** = (**expression**) returns and removes a stream of data from the receive register.

Object.**InputLen** = (**value**) sets and returns the number of characters the **Input** property reads from the receive buffer.

Object.**InputMode** = (**value**) sets and receives the type of data by the **Input** property. A value = 1 denotes text style data while a value = 0 denotes binary data.

Object.**Name** returns the name used in code to identify a form, control, or data access object.

Object.**NullDisregard** = (**true or false**) where the null characters are transferred from the port to the receive buffer when false and not transferred when true.

Object.**Object[.property|.method]** returns the object and/or setting of an object's method or property in an **OLE** container control.

Private Sub object_ **OnComm** () is generated whenever the value of the **CommEvent** property changes which indicates either a communication event or error has taken place.

Object.**OutBufferCount** = (**value**) returns the number of characters waiting in the transmit buffer or clears the buffer by setting the value = 0.

Object.**OutBufferSize** = (**value**) sets and returns the size of the transmit buffer in bytes. The default size is 512 bytes.

Object.**Output** = (**value**) transmits a stream of data to the transmit buffer. It can transmit text by specifying a **Variant** which contains a string, or binary data by passing a **Variant** which contains a byte array.

Object.**Parent** = (**value**) returns the collection, form, or object which contains a control or other object or collection.

Object.**ParityReplace** = (**value**) sets and returns the character that replaces an invalid character when a parity error occurs in a data stream.

Object.**Rthreshold** = (**value**) sets and returns the number of characters before the **CommEvent** property is set.

Object.**RTSEnable** = (**true or false**) controls the state of the Request To Send (RTS-pin 4 of DB-25) line as high when the value is true and low when the value is false.

Object.**Sthreshold** = (**value**) sets and returns the minimum number of characters in the transmit buffer before the **CommEvent** property is set.

Object.**Tag** = (**expression**) allows you to assign an ID string to an object without affecting any other properties of that object. The value isn't used by Visual Basic and is used strictly to identify an object.

Now, I know that's a lot of info in a few short pages, so I'm going to give you some time to digest it and understand how these properties work together. Next time, we'll begin to build a simple communication program to interface to our network. We have currently designed the interface using a MAX232 chip and a 75176 chip. The unit is half-duplex and is controlled via the RTS line. This is a proven generic design used extensively in the industry. Even though these aren't programmable chips, we'll be selling a semiconductor package of these two chips and a 78L05 voltage regulator package for \$ 10.00 including postage and handling in the continental USA . You supply the four 0.1 uf. capacitors and either a DB-9 or DB-25 connector.