I/O ActiveX Communications Control Instruction Manual

Copyright (c) 1998 JS Payne

Contents

Introduction

An ActiveX control is a software component that can easily be used in a variety of “Visual” programming environments. To use an ActiveX control, it must first be installed on the system being used for development. Then it can be inserted into the programming environment where it will be used. See “Inserting the Control”.

ActiveX controls are typically inserted on a form. After the control is placed on the form the member functions and properties are available to be used by the programmer. Also, placing the control on the form generates event functions that will be called when the control has an event to send to the application. Such events occur when:

- A change in the number of characters in the input or output buffers occurs.
- An error occurs.
- A job is finished.
- A change in status.

Before doing communication to a device connected to a port you must “open” the port (See Open()). When doing serial communication, you will need to setup the type of handshaking that your device uses (See SetHandshaking()). Now you can write data to the device through the opened port. You can also read data from the device using a read command. Often a request for data is written to the device on the port before a read is executed. When done talking to the device, the port is closed by calling the Close() function.

Some typical devices this control is used for communicating with are:

- Scanners.
- Scales.
- Other PCs (Serial only).
- Printers.
- Display Devices.
- Hardware Devices.
- Credit Card Readers.
- Temperature Measurement Devices.
- Bar Code Readers.
- Label Printers.
- Flow Measurement Devices.
- Check Readers.
- Load Cells.
- Pressure Measurement Devices.

Inserting the ActiveX control into your application.

Typically you will have an option to "insert control", "custom control", "OLE control", “ActiveX Control”, “More Controls” or "insert component".

You may have to select the control before it is available to you for insertion on your form. To do this, look for "I/O ActiveX Communications Module" or “I/O Control” and select the control.

Now you typically have the I/O control icon on your control palette. Select it and place it on your form.

Now you will have access to the I/O control, you can call the member functions and set the properties as meets your requirements.

In many environments, the control is named "IO1" and the functions and properties are accessed as per the following "IO1.Open("LPT1:", "")". Your environment may differ somewhat but the principles are the same. Look in the help for your programming environment for more information.

Some Examples:

I/O ActiveX Control in Visual Basic
1) Right click on the controls palette.
2) Select custom controls.
3) Check I/O control in list box.
4) Select OK.
5) Select the "I/O" control from the control palette and click on the Visual Basic form you are using.

I/O ActiveX Control in Access
1) Select “More Controls” on the controls palette.
2) Select I/O control in list box.
3) Place the control on the form you are using.

I/O ActiveX Control in Visual FoxPro
1.From the menu: "View" -> "Form Controls Toolbar".
2.Select "OLE Container Control" on Form Controls Toolbar.
3.Drop control onto the form.
4.Select "Insert Control".
5.Scroll down and select "IO Control".
6.Click "OK".
<u>OR:</u>
1.From the menu: "Tools", "Options", "Controls".
2.Select "ActiveX Controls".
3.Scroll down and select "IO Control".
4.Click "OK".
5.On "Forms Control Toolbar", "View Classes" -> "ActiveX".
6.Select "IO Control" from the "Forms Control Toolbar" and drop it on the form.

Copyright (c) 1998 by JS Payne

I/O ActiveX Control Guidelines

1. The Open function can fail on NT if another device is setup to use the port that is trying to be opened. Remove other drivers/devices from the port you are trying to open, including printer drivers and network re-directs.
2. When reading a parallel port the I/O ActiveX control will use the RS1284 protocol to read the data from the port.
3. The recommended serial cable is a "Null modem" cable.
4. You can connect two PCs together with a serial cable and send data to/from the connected PCs. Connect the PCs and run the IODemo on both PCs and select the correct ports and do reads and writes. You can run a terminal program on one PC in place of the IODemo.
5. How often status is monitored before an event is fired is controlled by the property StatusEventInterval. The default value is 250 milliseconds. Making this value smaller makes status events more sensitive to a change. A value of 0 disables the status events.
6. ParallelStatus can fail on Win NT if there is a printer driver setup to use the parallel port that I/O is being done on. Remove other drivers/devices from the port you are trying to use, including printer drivers.
7. For a more responsive user interface during I/O, consider using the Background mode of operation. Set the Mode property to MODE_ASYNC (2).
8. During application shutdown, errors can occur. To address this issue, ensure no calls to the I/O object are made after the I/O object is destroyed. To do this, move the I/O object shutdown processing to be done before the application shutdown processing is done. Note: It is recommended that "IO.Close()" is called before an application begins to shut down.
9. To clear the input and output buffers, Call the "IO.Close()" function and then reopen the port via the "IO.Open()" function.
10. Events are fired from the primary thread. If an event does not fire, call the IO.Wait() function with a value of 100 to 1000.
11. On Win NT, the parallel port does not accept timeouts. To avoid a dead lock, check the port status (ParallelStatus) before doing I/O on a parallel port on Win NT.

Redistribution of files:

1) You will need to redistribute "IO.OCX" with the application that you write.
2) "IO.OCX" may need to be registered with the OS when it is distributed.
    a) If you distribute with "InstallShield(R) Express 2" the IO.OCX will self register.
    b) If you distribute with Visual Basic setup utility the IO.OCX will self register.
    c) Other method: run "Regsvr32.exe IO.OCX".
3) OR copy IO.OCX to the "system" directory under "windows".

Files you may redistribute:

IO.OCX
MFC42.DLL
MSVCRT.DLL
OLEAUT32.DLL

Note: IO.OCX is dependent on MFC42.DLL, MSVCRT.DLL, and OLEAUT32.DLL. You will need to install these files in the Windows system directory.

Copyright (c) 1998 JS Payne

IO ActiveX Communication Module Interface

Communications

MAIN FUNCTIONS:
Open(PortName, Setup)
Close()
WriteString(Data)
ReadString(Length)
WriteByte(Data)
ReadByte()
SetTimeOut(Time)
SetTimeOuts(BaseTime, Multiplier) New
SetHandshaking(HSMethod)
WriteData(Data, Length)
ReadData(Length)

ADDITIONAL FUNCTIONS:
BytesRead()
ListPorts(Index, Type)
NumberRetries(Retries)
DataBuffer
Mode
PortName
NumBytesRead
NumBytesWritten
GetPortHandle()
SetBufferSize(InSize, OutSize)
Sleep()
Wait() New
OpenEx() New
CancelIO() New

Status

ParallelStatus
SerialStatus
NumCharsInQue
NumCharsOutQue

Communications Events

IOStatusEvent(StatusType, IOStatus)
IOCompleteEvent(JobType, JobId, JobResult)
IOQueueEvent(NumCharsInputQue, NumCharsOutputQue) New
IOPeriodicEvent() New
StatusEventInterval
PeriodicEventEnabled New

Parallel Specific

Serial Specific

Serial Specific (Advanced)

Open(PortName, Setup)

Parameters
PortName is the name of the port to be opened. Non standard port names and ports higher than 9, may need to be prefixed with "\\.\" and have no trailing ':' (in C/C++ "\\\\.\\").
Setup is a string that will set the mode of operation for a serial port. Not used with parallel ports.
Remarks
Opens a port for doing input and output.
Returns
1 if successful and 0 if the function fails.
2 if successful but the port setup fails can happen when another app is using the port.
Example
Result = IO1.Open("LPT1:", "") 'Open a parallel Port.
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
Result = IO1.Open("\\.\digi1", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
Result = IO1.Open("\\.\COM22", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
NOTE: On Win NT Open() can fail if another device/driver is setup to use the port you are trying to open. Remove all devices/drivers setup to use this port, including printer drivers.

Close()

Parameters
None.
Remarks
Closes an open port.
Returns
1 if successful and 0 if the function fails.
Example
Result = IO1.Close()

WriteString(Data)

Parameters
Data Thestring to be written
Remarks
Writes a string to the previously opened port. For binary data use the *WriteData* function. If the function fails, you may need to increase the timeout value via SetTimeOuts.
Returns
1 if successful and a 0 if the function fails.
Example
Result = IO1.WriteString("Hello World" + Chr(13) + Chr(10)) 'Sends "Hello World" to a printer or other device.

ReadString(Length)

Parameters
Length The number of characters to read from the port.
Remarks
Reads characters from an opened port and returns them as a string. For binary data use the function *ReadData*. When reading data from a parallel port, this function uses the RS1284 protocol for reading data from the parallel port. If the function fails, you may need to increase the timeout value via SetTimeOuts.
Example
String = IO1.ReadString(30) 'Reads data from a port. Typically a request for data precedes this command.

WriteByte(Data)

Parameters
Data The byte/character to be written
Remarks
Writes a byte/character to the previously opened port. If the function fails, you may need to increase the timeout value via SetTimeOuts.
Returns
1 if successful and a 0 if the function fails.
Example
Result = IO1.WriteByte(Chr(10)) 'Sends Line feed to a printer or other device.

ReadByte()

Parameters
None.
Remarks
Reads a byte/character from an opened port. When reading data from a parallel port, this function uses the RS1284 protocol for reading data from the parallel port. This function returns the byte data in an integer. Some environments my sign extend this value. You may wish to do a bitwise AND with the value read from this function and a hex FF to restore it to an unsigned value.
Returns
The byte/character.
Example
Result = IO1.ReadByte() 'Reads 1 data byte from a port. Typically a request for data precedes this command.
OR
Result = IO1.ReadByte() And &Hff 'restores to unsigned byte

SetTimeOut(Time)

Parameters
Time The base time in milliseconds. For Reading a port the timeout will be: Time + (number of characters to read) x Time .For writing to a port the time out will be: Time + (number of characters to write) x Time. A value of zero indicates that timeout are not used. Default timeout factor is 20.
Remarks
Sets the time out factor for a port. The write/read operation will be tried until the time is expired. If the timeout expires, the function will return an error (0) result.
Returns
1 if successful and a 0 if the function fails.
Example
Result = IO1.SetTimeOut(20) 'Sets timeout factor, how long a request is tried before an error is returned.

SetTimeOuts(BaseTime, Multiplier)

Parameters
BaseTime The base time in milliseconds.
Multiplier The multiplication time in milliseconds.
Remarks
Sets the time out factor for a port. The write/read operation will be tried until the time is expired. If the timeout expires, the function will return an error (0) result. For Reading a port the timeout will be: BaseTime + (number of characters to read) x Multiplier .For writing to a port the time out will be: BaseTime + (number of characters to write) x Multiplier.
Returns
1 if successful and a 0 if the function fails.
Example
Result = IO1.SetTimeOut(200, 20) 'Sets timeout factor, how long a request is tried before an error is returned.

SetHandshaking(HSMethod)

Parameters
HSMethod The type of handshaking protocol to be used with a serial port. 'Serial ports Only. 0 = None, 1 = Xon/Xoff, 2 = Hardware
Remarks
Sets the type of handshaking to be used with this port, the device must be configured to use the same type of handshaking protocol. Calling this function will reset the baud rate, parity, and stop bits to the values passed into the Open() function. This function should be called before the "Advanced Serial Communications" functions. SetHandshaking() will cancel the "Advanced Serial Communications" functions settings.
Returns
1 if successful and a 0 if the function fails.
Example
Result = IO1.SetHandshaking(2) 'Serial ports Only. 0 = None, 1 = Xon/Xoff, 2 = Hardware

WriteData(Data, Length)

Parameters
Data The binary data to be written, can include embedded nulls.
Length The length of data to be written.
Remarks
Writes data to the previously opened port. If the function fails, you may need to increase the timeout value via SetTimeOuts.
Returns
The length of data written if successful and a 0 if the function fails.
Example
Result = IO1.WriteData("String1" + Chr(00) + "String2" + Chr(00) + "String3" + Chr(00) + Chr(00))
'Sends 3 null terminated strings to the port, with the total data being double null terminated.

ReadData(Length)

Parameters
Length The number of bytes to read from the port.
Remarks
Reads bytes from an opened port and returns them in a string. The data can have null bytes. Even thought this function will return a string with embedded nulls other functions that operate on strings may not be null friendly. When reading data from a parallel port, this function uses the RS1284 protocol for reading data from the parallel port.
Returns
Binary data in a string.
Example
String = IO1.ReadData(30) 'Reads data from a port. Typically a request for data precedes this command.

BytesRead()

Parameters
None.
Remarks
Return the number of bytes previously read from the port. This value corresponds to the last read function done on this port. For use with the *ReadData()* function.
Returns
The number of bytes previously read function called.
Example
String = IO1.ReadData(30) 'Reads data from a port.
NumBytes = IO1.BytesRead() 'returns number of bytes read.

ListPorts(Index, Type)

Parameters
Index The nth item in the list to retrieve.
Type The type of port to be retrieve.
Remarks
Type is 1 for COM ports, 2 for LPT ports and, 4 for all ports. Note you can OR these values together to get a combination.
Returns
Returns the ports availble on the machine for the given index.
Example

```
For i = 0 To 10
Label2.Caption = Label2.Caption + IO1.ListPorts(i, 1) + " "
Next i
```

NumberRetries(Retries)

Parameters
Retries The number of times an I/O operation will be tried before a failure is returned.
Remarks
Default is 1 retry.
Returns
Returns 1.
Example

```
Result = IO1.NumberRetries(1)
```

DataBuffer (Property) R

Remarks
This buffer reflects the last read data. It is typically used during a IOCompleteEvent when a background read is completed. It's contents are not valid until a read has beed done (ie. ReadString).

Mode (Property) R/W

Remarks
Set this to 0 (MODE_NORMAL) for normal operation, or to 2 (MODE_ASYNC) for background I/O operations. In the background (or asynchronous) mode of operation, the IO control will perform read and write operations on a background process. Write operations will return a "Job ID" number. When the operation is complete, an IOComplete event will be fired signaling the completion of the operation and the result of the operation. This background operation will allow slow I/O operations to be done and not tie up the main application. When a background read function finishes the data will be in DataBuffer.

PortName (Property) R

Remarks
PortName reflects the name of the open port. This is provided for the convience of the application, the application upon detecting an error condition can use PortName to display a message to the application user to take corrective action.

NumBytesRead (Property) R

Remarks
This reflects the number of byte/characters read during the last read operation.

NumBytesWritten (Property) R

Remarks
This reflects the number of byte/characters write during the last read operation.

GetPortHandle()

Parameters
None .
Remarks
This function returns the handle of the currently opened port. This function is provided to allow calls to the Windows API directly.
Returns
The handle of the currently opened port.
Example
Result = IO1.GetPortHandle()

SetBufferSize(InSize, OutSize)

Parameters
InSize. Specifies the size of the input buffer.
OutSize. Specifies the size of the output buffer.
Remarks
This function sets the size of the input and output buffer associated with the opened port. Caution should be taken when using this function, some hardware running Windows 95 can function incorrectly when overriding the default buffer size with a serial port.
Returns
0 if the function fails.
Example
Result = IO1.SetbufferSize(InSize, OutSize)

Sleep(NumMilliseconds)

Parameters
NumMilliseconds. Specifies the length of the delay produced by calling this function.
Remarks
This function delays for the number of milliseconds specified. This function is provided to allow for custom timing of signals on the ports.
Returns
None.
Example
IO1.Sleep(500) ‘delay for 500ms (½ second).

Wait(NumMilliseconds)

Parameters
NumMilliseconds. Specifies the minimum length of the delay produced by calling this function.
Remarks
This function delays for the number of milliseconds specified. This function differs from Sleep() in that it will yield the current thread to allow the application to process user input. This yielding lets the application respond to user input instead of appearing to "hang" during a long delay. Because this functions yields to other tasks, it may delay for longer than the time specified.
Returns
None.
Example
IO1.Wait(500) ‘delay for at least 500ms (½ second).

OpenEx(PortName, Setup, Mode, Extra)

Parameters
PortName is the name of the port to be opened. Non standard port names and ports higher than 9, may need to be prefixed with "\\.\" and have no trailing ':' (in C/C++ "\\\\.\\").
Setup is a string that will set the mode of operation for a serial port. Not used with parallel ports.
Mode is the extra mode parameter dictating how the open is done:
OPEN_MODE_NORMAL (0) Extra = Not used
OPEN_MODE_SHARE (1) Extra = Not used
OPEN_MODE_BYHANDLE (2) Extra = Handle
Extra is extra information passed to the OpenEx() function as needed. See the Mode parameter.
Remarks
Opens a port for doing input and output.
OPEN_MODE_NORMAL Same as Open().
OPEN_MODE_SHARE Opens the port in a shared mode. This allows the port to be opened and used even if another applicaion/driver is using this port. This is most useful when using Win NT.
OPEN_MODE_BYHANDLE This allows the application writer to open the port and pass the handle to the I/O control.
Returns
1 if successful and 0 if the function fails.
2 if successful but the port setup fails can happen when another app/device driver is using the port.
Example
Result = IO1.Open("LPT1:", "", OPEN_MODE_SHARE, 0) 'Open a parallel Port. In shared mode.
NOTE: On Win NT Open() can fail if another device/driver is setup to use the port you are trying to open. Remove all devices/drivers setup to use this port, including printer drivers.

CancelIO(CancelFlags)

Parameters

CancelFlags Determines how and what cancel action is taken. These values can be added or ORed to combine the flags.

CANCEL_TXABORT (1) Abort the pending/current writes to the comm port.
CANCEL_RXABORT (2) Abort the pending/current reads from the comm port.
CANCEL_TXCLEAR (4) Purge the transmit queue.
CANCEL_RXCLEAR (8) Purge the receive buffer.

Remarks

Cancels pending reads or writes to the open port, also can purge/remove any information in the read or write buffers. Canceling pending I/O works best if using the background I/O Mode.

Returns
1 if successful and a 0 if the function fails.
Example
Result = IO1.CancelIO(1+2+4+8) 'Cancels pending I/O.

Status

ParallelStatus (Property) R

Remarks
This reflects the current status of the parallel/printer port as follows:
Example:

```
If (IOStatus And PARALLEL_SELECTED) Then
NewText = NewText + "Selected. "
Else
NewText = NewText + "Not Selected. "
End If
```

PARALLEL_PAPER_EMPTY	0x4
PARALLEL_OFF_LINE	0x8
PARALLEL_POWER_OFF	0x10
PARALLEL_NOT_CONNECTED	0x20
PARALLEL_BUSY	0x40
PARALLEL_SELECTED	0x80

SerialStatus (Property) R

Remarks
This reflects the current status of the serial port as follows:
Example:

```
If (IOStatus And SERIAL_RXEMPTY) Then
NewText = NewText + "RX Buffer Empty, "
else
NewText = NewText + "RX Buffer Not Empty, "
End If
```

SERIAL_RXOVER	0x0001	An input buffer overflow has occurred. There is either no room in the input buffer or a character was received after the end-of-file (EOF) character.
SERIAL_OVERRUN	0x0002	A character-buffer overrun has occurred. The next character is lost.
SERIAL_RXPARITY	0x0004	The hardware detected a parity error.
SERIAL_FRAME	0x0008	The hardware detected a framing error.
SERIAL_BREAK	0x0010	The hardware detected a break condition.
SERIAL_TXFULL	0x0100	The application tried to transmit a character, but the output buffer was full.
SERIAL_TXEMPTY	0x0020	The transmit buffer is empty.
SERIAL_RXEMPTY	0x0040	The receive buffer is empty.
SERIAL_CTS_TXHOLD	0x0200	Transmission is waiting for the CTS (clear-to-send) signal to be sent.
SERIAL_DSR_TXHOLD	0x0400	Transmission is waiting for the DSR (data-set-ready) signal to be sent.
SERIAL_RLSD_TXHOLD	0x0800	Transmission is waiting for the RLSD (receive-line-signal-detect) signal to be se
SERIAL_XOFF_TXHOLD	0x1000	Transmission is waiting because the XOFF character was received.
SERIAL_CTS_ON	0x010000	The CTS (clear-to-send) signal is on.
SERIAL_DSR_ON	0x020000	The DSR (data-set-ready) signal is on.
SERIAL_RING_ON	0x040000	The ring indicator signal is on.
SERIAL_RLSD_ON	0x080000	The RLSD (receive-line-signal-detect) signal is on.

NumCharsInQue (Property) R

Remarks
This reflects the number of bytes/characters in the input buffer (waiting to be read).

NumCharsOutQue (Property) R

Remarks
This reflects the number of bytes/characters in the output buffer (pending write to the port).

Events

IOStatusEvent(StatusType, IOStatus)

Parameters
StatusType. Specifies the type of status being reported.
STATUS_TYPE_PARALLEL 1
STATUS_TYPE_SERIAL 2
IOStatus. Specifies the status pertaining to the open port. See the definition of SerialStatusand ParallelStatusfor the value of this parameter.
Remarks
This event is fired whenever a status change is detected on the port. How often status is checked and events are generated is set by StatusEventInterval (default of 250 milliseconds).

IOCompleteEvent(JobType, JobId, JobResult)

Parameters
JobType. Specifies the type of Job being reported as finished.
BKJOB_WRITE 3
BKJOB_READ 4
JobId. Specifies the Id of the job submitted for background processing.
JobResult. Specifies the result of the job submitted for background processing.
Remarks
This event is fired whenever a background job is finished. If a read operation is completed, DataBuffer will contain the result of the read operation.

IOQueueEvent(NumCharsInputQue, NumCharsOutputQue)

Parameters
NumCharsInputQue Reflects the number of characters in the port's input queue/buffer.
NumCharsOutputQue Reflects the number of characters in the port's output queue/buffer.
Remarks
This event is fired whenever the number of character in either the input or output queue has changed. This event is provided to allow the application designer to monitor the communications activity.
Example:

```
If (NumCharsInputQue > 0) Then
    TextRead.Text = TextRead.Text + IO1.ReadData(20)
End If
```

IOPeriodicEvent()

Parameters
None
Remarks
This event is fired whenever the StatusEventInterval has expired. This event is provided to allow the applications designer to add custom communications monitoring on a periodic basis. This event is enable by setting the property PeriodicEventEnabled to TRUE (1).

StatusEventInterval (Property) R/W

Remarks
Set this to control how often changes in port status are monitored. This affects how often IOStatusEvents, IOQueueEvents, and IOPeriodicEvents are fired and how responsive status events are to changes in port status. If this is set to 0, status events are not fired and the thread monitoring status for status events is terminated until this property is set to non-zero. The units for this property are in milliseconds. The default value is 250 milliseconds.

PeriodicEventEnabled (Property) R/W

Remarks
Set this to enable the IOPeriodicEvent.

Parallel Specific

InitPrinter(Code)

Parameters
Code. Specifies what action to take.
  1 assert init signal to device on printer port.
Remarks
Executes a printer reset via the Init pin on the parallel printer port.
Returns
1 for success and 0 if the function fails.

Example
ret = IO1.InitPrinter(1)

Out (Address, Data) Win95/98 Only

Parameters
Address. Specifies the address of the port hardware.
Data. Specifies the byte to be written.
Remarks
Because this is used to write a data byte directly to the parallel port hardware, do not call the Open() or Close() functions when using the Out() function.. See below for the address of standard parallel ports.
Port addresses:
LPT1: 378 Hex(Data), 379 Hex(Status), 37A Hex(Control)
LPT2: 278 Hex(Data), 279 Hex(Status), 27A Hex(Control)
LPT3: 3BC Hex(Data), 3BD Hex(Status), 3BE Hex(Control)

In (Address) Win95/98 Only

Parameters
Address. Specifies the address of the port hardware.
Remarks
Because this is used to read a data byte directly from the parallel port hardware, do not call the Open() or Close() functions when using the In() function. See below for the address of standard parallel ports.
Returns
The data byte of the addressed port register.
Port addresses:
LPT1: 378 Hex(Data), 379 Hex(Status), 37A Hex(Control)
LPT2: 278 Hex(Data), 279 Hex(Status), 27A Hex(Control)
LPT3: 3BC Hex(Data), 3BD Hex(Status), 3BE Hex(Control)

DeviceControl(Command, Data, Buffer) WinNT Only

Parameters
Command. Specifies the command.
    ASSERT_PAR_CONTROL (1441800) Assert parallel port control signal(s).
Data. Specifies data that the command uses.
    PARALLEL_INIT (1)
    PARALLEL_AUTOFEED (2)
Buffer. Specifies additional data as needed.
Remarks
Sends a command to port hardware through the port driver.
Returns
The return data is a string. If the function fails the string is empty. If the function succeeds the string is non-empty and its contents depend on the command sent.

Example
```
ret = IO1.DeviceControl(1441800, 1, "")
```

Serial Specific

SerialPortSetupDialog(PortName)

Parameters
PortName Specifies the port name used by the setup dialog.
Remarks
This method displays the serial port setup dialog and allows a user to adjust the port settings. After the dialog is dismissed by clicking on the "OK" button, the new settings are applied to the currently open port. If no port is open, the settings are saved in the I/O object and can be stored by calling the SerialSetPortDefaults() function. If the PortName is blank (i.e. ""), the dialog will use the port name of the most recently opened port.
Returns
1 for success
0 for fail or cancel button was clicked.
Example

```
IO1.SerialGetPortDefaults (PortName) 'Get port settings
Result = IO1.SerialPortSetupDialog(PortName) 'Allow user to setup port
If Result = 1 Then
Result = IO1.SerialSetPortDefaults(PortName, "", -1) 'Save the settings
End If
```

SerialSetPortDefaults(PortName, Setup, HSMode)

Parameters
PortName Specifies the port to set defaults.
Setup Is a setup string as described in Open()
HSMode Is the handshaking mode, see SetHandshaking()
Remarks
This method allows setting of the default settings of a serial port. The default settings for a port are maintained/save by the operating system. If Setup is blank (i.e. ""), this function will use the settings of the last opened port or the settings entered via the SerialPortSetupDialog() function. If the HSMode is -1, this function will use the settings of the last opened port or the settings entered via the SerialPortSetupDialog() function.. If the PortName is blank (i.e. ""), this function will use the port name of the most recently opened port. This function will also change/set the settings for the currently opened port (if one is opened).
Returns
1 for success or 0 for fail.
Example

```
IO1.SerialPortSetupDialog (PortName) 'Allow user to setup port
Result = IO1.SerialSetPortDefaults(PortName, "", -1) 'Save the settings
```

SerialGetPortDefaults(PortName)

Parameters
PortName Specifies the port to get default settings.
Remarks
This method allows retrieval of the default settings of a serial port. The default setting for a port are maintained/save by the operating system. If the PortName is blank (i.e. ""), this function will use the port name of the most recently opened port. This function should be called after the port is opened, as opening of the port will overwrite the setting retrieved by calling this function.
Returns
1 for success or 0 for fail.
Example
IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open port with initial settings
IO1.SerialGetPortDefaults (PortName) 'Get port settings previously stored by SerialSetPortDefaults

SerialBreak(BreakMode)

Parameters
BreakMode Specifies the state of the serial port break.
Remarks
This method allows the open serial port to be put into a break condition.
Returns
1 for success or 0 for fail.
Example
IO1.SerialBreak (1) 'Set serial port into a break condition.
IO1.Sleep(100)
IO1.SerialBreak (0) 'Set serial port into a non-break condition.

Advanced Serial Communications functions

These functions are not typically needed to successfully do serial communication, but are provided to allow advanced control of the serial port. These functions should be called after the *SetHandshaking()* function is called, as *SetHandshaking()* will cancel these settings.

SerialCTSFlow(Value)

Parameters
Value 1 or 0. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether the CTS (clear-to-send) signal is monitored for output flow control. If it is 1 and CTS is low, output is suspended until CTS is high again. The CTS signal is under control of the device (usually a modem/printer), the host simply monitors the status of this signal, the host does not change it. This function should be called before the port is opened.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialCTSFlow(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialDSRFlow(Value)

Parameters
Value 1 or 0. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether the DSR (data-set-ready) signal is monitored for output flow control. If this member is 1 and DSR is low, output is suspended until DSR is high again. This signal is under the control of the attached device; the host(PC) only monitors this signal. This function should be called before the port is opened.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialDSRFlow(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialDTRControl(long Value)

Parameters
Value 0,1,2,5,6. Controls state of DTR line. Undetermined results if set to other than specified
Remarks
If 0 then the DTR line is lowered when the device is opened.
If 1 then the DTR line is raised when the device is opened.
If set to 2 enables DTR flow-control handshaking.
<u>After the port is opened:</u>
If called with a value of 5 sets DTR high. *Only allowed if SerialDTRControl was set to 1 0r 0 before the port was opened.*
If called with a value of 6 sets DTR low. *Only allowed if SerialDTRControl was set to 1 0r 0 before the port was opened.*
Returns
1 if successful or 0 if failed.
Example
Result = IO1.SerialDTRControl(0)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
...
Result = IO1.SerialDTRControl(5)

SerialDSRSensitivity(Value)

Parameters
Value 1 or 0. Controls DSR Sensitivity. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether the communications driver is sensitive to the state of the DSR signal. If this is set to a 1, the driver ignores any bytes received, unless the DSR modem input line is high.
Returns
1 if successful or 0 if failed.
Example
Result = IO1.SerialDSRSensitivity(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.

SerialTxContinueOnXoff(Value)

Parameters
Value 1 or 0. Controls Xon/Xoff behavior. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether transmission stops when the input buffer is full and the driver has transmitted the XOFF character. If this is set to a 1, transmission continues after the XOFF character has been sent. If this is set to a 0, transmission does not continue until the input buffer is within SerialXonLim bytes of being empty and the driver has transmitted the XON character.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialTxContinueOnXoff(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialOutX(Value)

Parameters
Value 1 or 0. Controls Xon/Xoff behavior. Undetermined results if set to other than 1 or 0
Remarks
Specifies whether XON/XOFF flow control is used during transmission. If this member is a 1, transmission stops when the XOFF character is received and starts again when the XON character is received.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialOutX(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialInX(Value)

Parameters
Value 1 or 0. Controls Xon/Xoff behavior. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether XON/XOFF flow control is used during reception. If this is set to a 1, the XOFF character is sent when the input buffer comes within a preset limit of bytes of being full (see *SerialXonLimit* and *SerialXoffLimit*), and the XON character is sent when the input buffer comes within a preset limit of bytes of being empty.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialInX(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialErrorReplacment(Value)

Parameters
Value 1 or 0. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether bytes received with parity errors are replaced with the character specified by the *SerialErrorCharacter* member function. If this is set to a 1 and the Parity is set to a 1, replacement occurs.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialErrorReplacment(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialNullStripping(Value)

Parameters
Value 1 or 0. Undetermined results if set to other than 1 or 0.
Remarks
Specifies whether null bytes are discarded. If this is set to 1, null bytes are discarded when received.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialNullStripping(1)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialRTSControl(Value)

Parameters
Value 0,1,2,3,5,6. Controls state of RTS line. Undetermined results if set to other than specified
Remarks
If 0 then the RTS line is lowered when the device is opened.
If 1 then the RTS line is raised when the device is opened.
If set to 2 enables RTS flow-control handshaking. The driver raises the RTS line, enabling the attached device to send (when the input buffer has enough room to receive data). The driver lowers the RTS line, preventing the attached device from sending (when the input buffer does not have enough room to receive data).
If set to 3 specifies that the RTS line will be high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line will be low. (Not supported on Win95)
<u>After the port is opened:</u>
If called with a value of 5 sets RTS high. *Only allowed if SerialRTSControl was set to 1 0r 0 before the port was opened.*
If called with a value of 6 sets RTS low. *Only allowed if SerialRTSControl was set to 1 0r 0 before the port was opened.*
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialRTSControl(0)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
...
Result = IO1.SerialRTSControl(5)
```

SerialXonLimit(Value)

Parameters
Value Xon Limit.
Remarks
Specifies the minimum number of bytes allowed in the input buffer before the XON character is sent.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialXonLimit(256)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialXoffLimit(Value)

Parameters
Value Xoff Limit.
Remarks
Specifies the maximum number of bytes allowed in the input buffer before the XOFF character is sent.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialXoffLimit(100)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialXonCharacter(Value)

Parameters
Value Xon character.
Remarks
Specifies the Xon character.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialXonCharacter(17)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialXoffCharacter(Value)

Parameters
Value Xoff character.
Remarks
Specifies the Xoff character.
Returns
1 if successful or 0 if failed.
Example

```
Result = IO1.SerialXoffCharacter(19)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialErrorCharacter(Value)

Parameters
Value Error character.
Remarks
Specifies the Error character, the character used to replace bytes received with a parity error.
Returns
1 if successful or 0 if failed.
Example
```
Result = IO1.SerialErrorCharacter('?')
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

SerialEndCharacter(Value)

Parameters
Value End character.
Remarks
Specifies the End character, the character used to signal the end of data.
Returns
1 if successful or 0 if failed.
Example
```
Result = IO1.SerialEndCharacter(FF)
...
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1") 'Open a serial Port.
```

Copyright (c) 1998 by JS Payne

Instructions for serial communication using I/O ActiveX Control

Open the port:
Result = IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1")

"COM2:" is the name of the port.
"baud=9600" specifies how fast the data is sent.*
"parity=N" specifies an error detection protocol.*
"data=8" specifies how many data bits are sent in a packet.*
"stop=1" specifies how many bits terminate a packet.*

Set type of handshaking:
Result = IO1.SetHandshaking(HSMethod)
Where 0 = None, 1 = Xon/Xoff, 2 = Hardware*

Handshaking is a method of "holding off" or "preventing" communication when a device is not ready to receive data.

Do the I/O:
Result = IO1.WriteString("Hello World" + Chr(13) + Chr(10)) 'Sends "Hello World" to a printer or other device.
String = IO1.ReadString(30) 'Reads data from a port. Typically a request for data precedes this command.

Close the port:
Result = IO1.Close()

*NOTE: These settings MUST match the settings of the device you are communicating with.

Manual Manipulation of Serial Communication Handshake Lines.

This ActiveX control can be used to manually manipulate the hardware handshaking lines of a serial communications port. To do this, follow the following steps.

1. Set the state of the control line before the port is opened.
2. Open the port.
3. Then assert the control line to the state you wish.

RTS Example:

```
IO1.SerialRTSControl(1) 'or 0 see SerialRTSControl documentation in this document.
IO1.Open("COM2:", "baud=9600 parity=N data=8 stop=1")
IO1.SerialRTSControl(5) 'or 6 to change RTS state as desired.
```

See other serial functions for other handshaking lines.

I/O OCX Control for Binary Data Transfers

There are three functions intended for binary data transfers.
1) WriteData(Data, Length); Writes a data buffer of Length characters.
2) ReadData(Length); Reads binary data and returns it in a string.
3) BytesRead(); Returns the number of bytes read by the ReadData() function.

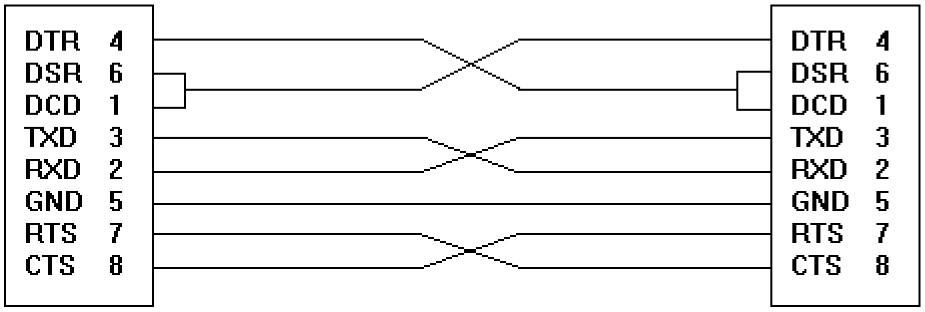
Example:

```
Dim Result as Integer
Dim DataStr as String
Dim NumBytes as Integer

Result = IO1.WriteData("String1" + Chr(00) + "String2" + Chr(00) + "String3" + Chr(00) + Chr(00))
'Sends 3 null terminated strings to the port, with the total data being double null terminated.

DataStr = IO1.ReadData(30) 'Reads data from a port.

NumBytes = IO1.BytesRead() 'returns number of bytes read.
```

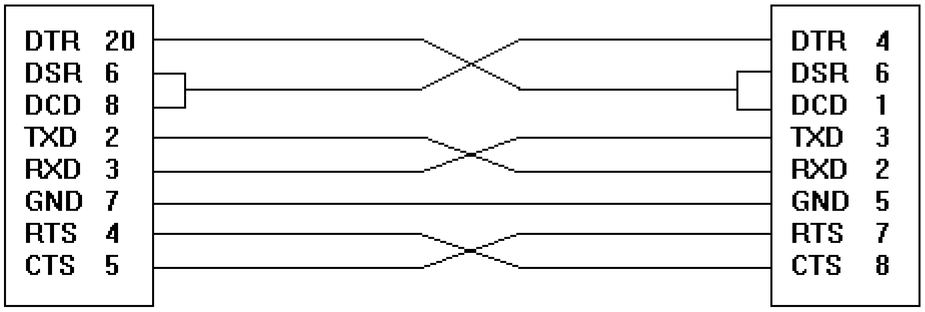
See WriteData, ReadData, and BytesRead in I/O OCX Control Functions

Serial and Parallel cable schematics and wiring diagrams.

9 PIN to 9 PIN Serial Cable



25 PIN to 9 PIN Serial Cable



Parallel Printer Connector Db-25
DB-25 PIN (Female) SIGNAL
DB-25 MALE CONN DB-25 FEMALE CONN
1 ------------------------------ > STROBE *
2 ------------------------------ > DATA 0
3 ------------------------------ > DATA 1
4 ------------------------------ > DATA 2
5 ------------------------------ > DATA 3
6 ------------------------------ > DATA 4
7 ------------------------------ > DATA 5
8 ------------------------------ > DATA 6
9 ------------------------------ > DATA 7
10< ------------------------------ ACK *
11< ------------------------------ BUSY
12< ------------------------------ PAPER END
13 ------------------------------ SLCT (select)
14 ------------------------------ >AUTOFEED *
15< ------------------------------ ERROR *
16 -------------------------->INITIALIZE PRINTER *
17 ------------------------------- SLCTIN (select in)
18 thru 25 ---------------------- GND
Note!! * denotes an active low signal.

Db-25 Parallel Loopback Connector Wiring
1 to 13 Strobe to select
10 to 16 ACK to INIT
11 to 17 BUSY to SLCTIN
12 to 20 PAPER END to GND

Rs-232 Serial (Com) Pc Port Connector Db-9
DB-9 PIN (Male) FUNCTION ABBREVIATION
1 -------------------------- Data Carrier Detect CD or DCD
2 ------------------------------ Receive Data RD or RX
3 ---------------------------- Transmitted Data TX or TD
4 ---------------------------- Data Terminal Ready DTR
5 ------------------------------ Signal Ground GND
6 ------------------------------ Data Set Ready DSR
7 ------------------------------ Request To Send RTS
8 ------------------------------ Clear To Send CTS
9 ------------------------------ Ring Indicator RI
Transmitted and receive data are referenced from the data device and not the modem.

Rs-232 Serial (Com) Pc Port Connector Db-25
DB-25 PIN (Male) FUNCTION ABBREVIATION
1 ---------------------------- Chassis/Frame Ground GND
2 ------------------------------ Transmitted Data TX or TD
3 -------------------------------- Receive Data RX or RD
4 ------------------------------ Request To Send RTS
5 ------------------------------- Clear To Send CTS
6 ------------------------------- Data Set Ready DSR
7 ------------------------------- Signal Ground GND
8 ---------------------------- Data Carrier Detect DCD or CD
9 ------------------------- Transmit + (Current loop) TD+
11 ----------------------- Transmit - (Current Loop) TD-
18 ------------------------- Receive + (Current Loop) RD+
20 --------------------------- Data Terminal Ready DTR
22 ----------------------------- Ring Indicator RI
25 ------------------------- Receive - (Current Loop) RD-
NOTE!! Current loop technology was supported in the PC and XT interfaces.
Current loop was discontinued when the AT interface was introduced.
Transmitted and receive data are referenced from the data
device and not the modem.

Db-25 Female Serial Loopback Plug Wiring
2 to 3 Xmit to Rec data
4 to 5 to 22 RTS to CTS to RI
6 to 8 to 20 DSR to CD to DTR
Db-9 Female Serial Loopback Plug Wiring
2 to 3 Xmit to Rec data
7 to 8 to 9 RTS to CTS to RI
6 to 1 to 4 DSR to CD to DTR

Rs-232 Serial Db-9 To Rs-232 Db-25 Adaptor
DB-9 PIN (Female) DB-25 PIN (Male)
1 ------------------------------------- 8 DCD
2 ------------------------------------- 3 TXD
3 ------------------------------------- 2 RXD
4 ------------------------------------- 20 DTR
5 ------------------------------------- 7 GND
6 ------------------------------------- 6 DSR
7 ------------------------------------- 4 RTS
8 ------------------------------------- 5 CTS
9 ------------------------------------- 22 RI
Use this pin out to adapt between the two serial connector
types.

Rs-232 Serial Db-25 To Db-25 Null Modem Cable
DB-25 PIN (Female) PC DB-25 PIN (Female) PC
2 ------------------------------------ 3
3 ------------------------------------ 2
7 ------------------------------------ 7
4 ------------------------------------ 5
5 ------------------------------------ 4
6 ------------------------------------ 20
20 ----------------------------------- 6
Note!! All other pins are unused. Use this cable pinout for direct connection between two IBM compatible computers.

Rs-232 Serial Db-25 To Serial Printer Null Modem Cable
DB-9 Female PC DB-25 PIN Female PC DB-25 PIN Male printer
2 < RD --------- 3 <------------------------------------ 2 Transmitted data
3 > TD --------->2 ------------------------------------> 3 Receive data
5 < GND -------- 7 <-----------------------------------> 7 Ground
8 < CTS -------- 5 ------------------------------------ 6 to 8 to 20
1 to 4 to 6 6 to 8 to 20 4 to 5
DTR/DSR/DCD DTR/DSR/DCD RTS to CTS
Note!! Use this cable pinout for direct connection between a PC serial port and a serial printer. The 1/4/6 and 6/8/20 loopbacks are to enable the interface as if a modem was attached.

Standard Centronics Parallel Cable Db-25 To Centronics 36
DB-25 PIN Male (PC) Centronics 36 Male
CENTRONICS 36 MALE CENTRONICS 36 FEMALE
1 -------------------------------------> 1 Strobe *
2 <------------------------------------> 2 Data bit 0 +
3 <------------------------------------> 3 Data bit 1 +
4 <------------------------------------> 4 Data bit 2 +
5 <------------------------------------> 6 Data bit 3 +
6 <------------------------------------> 6 Data bit 4+
7 <------------------------------------> 7 Data bit 5 +
8 <------------------------------------> 8 Data bit 6 +
9 <------------------------------------> 9 Data bit 7 +
10 <------------------------------------ 10 Acknowledge *
11 <------------------------------------ 11 Busy +
12 <------------------------------------ 12 Paper out +
13 <------------------------------------ 13 Select (in) *
14 ------------------------------------> 14 Auto Feed *
15 <------------------------------------ 32 Error *
16 ------------------------------------> 31 Initialize printer *
17 ------------------------------------> 36 Select (out) *
18 thru 25 Gnd 16, 19 thru 30, 33 Ground
15, 17, 18, 34, 35 No connection
Note!! * denotes and active low signal. This pin out depicts the newer bi-directional parallel port with input and output capabilities often used with external tape drives and accessory devices. If pins 31 or 32 are grounded on a cable the printer will fail to come ready when attached to the PC. This is common on low cost parallel printer cables.

Copyright (c) 1998 by JS Payne