Caller ID Software

Caller ID, MSCOMM32.OCX MSComm Control

Home
Hardware
Programming
Purchase
Contact

MSCOMM32.OCX MSComm Control

The MSComm control (MSCOMM32.OCX) comes with Visual Basic. This one came with VB 6.0. We have a license to distribute this OCX with our Visual Basic application. We have tried in to get a clarification from Microsoft if it is OK to distribute stand alone. We asked: "Do we have the right to distribute the MSComm control which comes with Visual Basic with an Access application?" When this question was asked of the MS Access technical support people we could not find anyone that knew what the MSComm control is. The closest we got was someone that thought we were talking about the Common Dialog control. It is our assumption that because we have the right to distribute this with a VB application we also should have the right to distribute it stand-alone.

This is the MSCOMM32.OCX that should be stored in your \windows\system directory.

To register within Access go to TOOLS/ACTIVE X CONTROLS and select \windows\system\mscomm32.ocx or Microsoft Communications Control and click on the REGISTER button.

If you get a licensing error the only way we know to get the licensing is to install VB 6 Professional. You can try the registry entry listed below. Microsoft Technical Support and Sales Staff was of no use when we tried find out how to register or purchase the Comm

control.

We were sent this string to be entered into the registry to clear the license issue. We have not tried it but the sender says it worked for them.

```
[HKEY_CLASSES_ROOT\Licenses\4250E830-6AC2-11cf-8ADB-00AA00C00905]
@ = "kjljvjjjoquqmjjjvpqqkqmqykypoqjquoun"
```

 **Download Now** mscomm.zip 50K bytes

The **MSComm** control provides serial communications for your application by allowing the transmission and reception of data through a serial port.
Syntax
MSComm
Remarks
The **MSComm** control provides the following two ways for handling communications:

- Event-driven communications is a very powerful method for handling serial port interactions. In many situations you want to be notified the moment an event takes place, such as when a character arrives or a change occurs in the Carrier Detect (CD) or Request To Send (RTS) lines. In such cases, use the **MSComm** control's **OnComm** event to trap and handle these communications events. The **OnComm** event also detects and handles communications errors. For a list of all possible events and communications errors, see the **CommEvent** property.
- You can also poll for events and errors by checking the value of the **CommEvent** property after each critical function of your program. This may be preferable if your application is small and self-contained. For example, if you are writing a simple phone dialer, it may not make sense to generate an event after receiving every character, because the only characters you plan to receive are the OK response from the modem.

Each **MSComm** control you use corresponds to one serial port. If you need to access more than one serial port in your application, you must use more than one **MSComm** control. The port address and interrupt address can be changed from the Windows Control Panel.
Although the **MSComm** control has many important properties, there are a few that you should be familiar with first.

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

OnComm Event

The **OnComm** event is generated whenever the value of the **CommEvent** property changes, indicating that either a communication event or an error occurred.
Syntax
Private Sub *object*_**OnComm ()**
The **OnComm** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Remarks
The **CommEvent** property contains the numeric code of the actual error or event that generated the **OnComm** event. Note that setting the **RThreshold** or **SThreshold** properties to 0 disables trapping for

the **comEvReceive** and **comEvSend** events, respectively.

CommPort Property

Sets and returns the communications port number.
Syntax
object.**CommPort**[= *value*]
The **CommPort** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A integer value specifying the port number.

Remarks
You can set *value* to any number between 1 and 16 at design time (the default is 1). However, the **MSComm** control generates error 68 (`Device unavailable`) if the port does not exist when you attempt to open it with the **PortOpen** property.
Warning You must set the **CommPort** property before opening the port.
Data Type
Integer

Handshaking Property

Sets and returns the hardware handshaking protocol.
Syntax
object.**Handshaking** [= *value*]
The **Handshaking** property syntax has these parts:

Part	Description

object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression specifying the handshaking protocol, as described in Settings.

Settings
The settings for *value* are:

Setting	Value	Description
comNone	0	(Default) No handshaking.
comXOnXOff	1	XON/XOFF handshaking.
comRTS	2	RTS/CTS (Request To Send/Clear To Send) handshaking.
comRTSXOnXOff	3	Both Request To Send and XON/XOFF handshaking.

Remarks
Handshaking refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. When a character of data arrives at the serial port, the communications device has to move it into the receive buffer so that your program can read it. If there is no receive buffer and your program is expected to read every character directly from the hardware, you will probably lose data because the characters can arrive very quickly.
A **handshaking** protocol insures data is not lost due to a buffer overrun, where data arrives at the port too quickly for the communications device to move the data into the receive buffer.
Data Type
Integer

RThreshold Property

Sets and returns the number of characters to receive before the

MSComm control sets the **CommEvent** property to **comEvReceive** and generates the **OnComm** event.

Syntax

object.**Rthreshold** [= *value*]

The **Rthreshold** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression specifying the number of characters to receive before generating the OnComm event.

Remarks

Setting the **RThreshold** property to 0 (the default) disables generating the **OnComm** event when characters are received. Setting **RThreshold** to 1, for example, causes the **MSComm** control to generate the **OnComm** event every time a single character is placed in the receive buffer.

Data Type

Integer

Settings Property

Sets and returns the baud rate, parity, data bit, and stop bit parameters.

Syntax

object.**Settings** [= *value*]

The **Settings** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An string expression representing the communications port settings, as described below.

Remarks
If *value* is not valid when the port is opened, the **MSComm** control generates error 380 (`Invalid property value`).
Value is composed of four settings and has the following format:

```
"BBBB,P,D,S"
```

Where `BBBB` is the baud rate, `P` is the parity, `D` is the number of data bits, and `S` is the number of stop bits. The default value of *value* is:

```
"9600,N,8,1"
```

The following table lists the valid baud rates.

Setting
110
300
600
1200
2400
9600 (Default)
14400
19200
28800
38400
56000
128000
256000

The following table describes the valid parity values.

Setting	Description
E	Even
M	Mark
N	(Default) None
O	Odd
S	Space

The following table lists the valid data bit values.

Setting
4
5
6
7
8 (Default)

The following table lists the valid stop bit values.

Setting	
1	(Default)
1.5	
2	

Data Type
String

InputLen Property

Sets and returns the number of characters the **Input** property reads

from the receive buffer.
Syntax
object.**InputLen** [= *value*]
The **InputLen** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression specifying the number of characters the **Input** property reads from the receive buffer.

Remarks
The default value for the **InputLen** property is 0. Setting **InputLen** to 0 causes the **MSComm** control to read the entire contents of the receive buffer when **Input** is used.
If **InputLen** characters are not available in the receive buffer, the **Input** property returns a zero-length string (""). The user can optionally check the **InBufferCount** property to determine if the required number of characters are present before using **Input**.
This property is useful when reading data from a machine whose output is formatted in fixed-length blocks of data.
Data Type
Integer

Input Property

Returns and removes a stream of data from the receive buffer. This property is not available at design time and is read-only at run time.
Syntax
object.**Input**
The **Input** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Remarks
The **InputLen** property determines the number of characters that are read by the **Input** property. Setting **InputLen** to 0 causes the **Input** property to read the entire contents of the receive buffer.
The **InputMode** property determines the type of data that is retrieved with the **Input** property. If **InputMode** is set to **comInputModeText** then the **Input** property returns text data in a **Variant**. If **InputMode** is **comInputModeBinary** then the **Input** property returns binary data in an array of bytes in a **Variant**.
Data Type
Variant

PortOpen Property

Sets and returns the state of the communications port (open or closed). Not available at design time.
Syntax
object.**PortOpen** [= *value*]
The **PortOpen** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A boolean expression specifying the state of the communications port.

Settings
The settings for *value* are:

Setting	Description
True	Port is opened
False	Port is closed

Remarks
Setting the **PortOpen** property to **True** opens the port. Setting it to

False closes the port and clears the receive and transmit buffers. The **MSComm** control automatically closes the serial port when your application is terminated.
Make sure the **CommPort** property is set to a valid port number before opening the port. If the **CommPort** property is set to an invalid port number when you try to open the port, the **MSComm** control generates error 68 `(Device unavailable)`.
In addition, your serial port device must support the current values in the **Settings** property. If the **Settings** property contains communications settings that your hardware does not support, your hardware may not work correctly.
If either the **DTREnable** or the **RTSEnable** properties is set to **True** before the port is opened, the properties are set to **False** when the port is closed. Otherwise, the DTR and RTS lines remain in their previous state.
Data Type
Boolean

DTREnable Property

Determines whether to enable the Data Terminal Ready (DTR) line during communications. Typically, the Data Terminal Ready signal is sent by a computer to its modem to indicate that the computer is ready to accept incoming transmission.
Syntax
object.**DTREnable**[= *value*]
The **DTREnable** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Value	A Boolean expression specifying whether to enable the Data Terminal Ready (DTR) line, as described in Settings.

Settings
The settings for *value* are:

Setting	Description
True	Enable the Data Terminal Ready line.
False	(Default) Disable the Data Terminal Ready line.

Remarks
When **DTREnable** is set to **True**, the Data Terminal Ready line is set to high (on) when the port is opened, and low (off) when the port is closed. When **DTREnable** is set to **False**, the Data Terminal Ready always remains low.
Note In most cases, setting the Data Terminal Ready line to low hangs up the telephone.
Data Type
Boolean

RTSEnable Property

Determines whether to enable the Request To Send (RTS) line. Typically, the Request To Send signal that requests permission to transmit data is sent from a computer to its attached modem.
Syntax
object.**RTSEnable**[= *value*]
The **RTSEnable** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An boolean expression specifying whether the Request To Send (RTS) line is enabled, as described in Settings.

Settings
The settings for *value* are:

Setting	Description
True	Enables the Request To Send line.

False	(Default) Disables the Request To Send line.

Remarks
When **RTSEnable** is set to **True**, the Request To Send line is set to high (on) when the port is opened, and low (off) when the port is closed.
The Request To Send line is used in RTS/CTS hardware handshaking. The **RTSEnable** property allows you to manually poll the Request To Send line if you need to determine its state.
For more information on handshaking protocols, see the **Handshaking** property.
Data Type
Boolean

CommEvent Property

Returns the most recent communication event or error. This property is not available at design time and is read-only at run time.
Syntax
object.**CommEvent**
The **CommEvent** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Remarks
Although the **OnComm** event is generated whenever a communication error or event occurs, the **CommEvent** property holds the numeric code for that error or event. To determine the actual error or event that caused the **OnComm** event, you must reference the **CommEvent** property.
The **CommEvent** property returns one of the following values for communication errors or events. These constants can also be found in the Object Library for this control.
Communication errors include the following settings:

Constant	Value	Description

comEventBreak	1001	A Break signal was received.
comEventFrame	1004	Framing Error. The hardware detected a framing error.
comEventOverrun	1006	Port Overrun. A character was not read from the hardware before the next character arrived and was lost.
comEventRxOver	1008	Receive Buffer Overflow. There is no room in the receive buffer.
comEventRxParity	1009	Parity Error. The hardware detected a parity error.
comEventTxFull	1010	Transmit Buffer Full. The transmit buffer was full while trying to queue a character.
comEventDCB	1011	Unexpected error retrieving Device Control Block (DCB) for the port.

Communications events include the following settings:

Constant	Value	Description
comEvSend	1	There are fewer than Sthreshold number of characters in the transmit buffer.
comEvReceive	2	Received Rthreshold number of characters. This event is generated continuously until you use the Input property to remove the data from the receive buffer.
comEvCTS	3	Change in Clear To Send line.
comEvDSR	4	Change in Data Set Ready line. This event is only fired when DSR changes from 1 to 0.
comEvCD	5	Change in Carrier Detect line.
comEvRing	6	Ring detected. Some UARTs (universal asynchronous receiver-transmitters) may not

		support this event.
comEvEOF	7	End Of File (ASCII character 26) character received.

Data Type
Integer

Output Property

Writes a stream of data to the transmit buffer. This property is not available at design time and is write-only at run time.

Syntax
object.**Output** [= *value*]
The **Output** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string of characters to write to the transmit buffer.

Remarks
The **Output** property can transmit text data or binary data. To send text data using the **Output** property, you must specify a **Variant** that contains a string. To send binary data, you must pass a **Variant** which contains a byte array to the **Output** property.
Normally, if you are sending an ANSI string to an application, you can send it as text data. If you have data that contains embedded control characters, Null characters, etc., then you will want to pass it as binary data.

Data Type
Variant