Contents

AutoProcess

Purpose:The AutoProcess property controls automatic processing of serial port and keyboard data.

Syntax: [form.]Comm1.AutoProcess[= integer%]

Design Time Interface: Drop-Down List Box
Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments:Valid settings are:

HDC_AUTOPROCESS_NONE	'-- None
HDC_AUTOPROCESS_SERIAL	'-- Serial Input
HDC_AUTOPROCESS_KEY	'-- Keyboard Input
HDC_AUTOPROCESS_BOTH	'-- Both Serial and Keyboard Input<DJ0>

When AutoProcess is set to HDC_AUTOPROCESS_NONE, HDCTerm will not process serial or keyboard input automatically. Even though AutoProcess is set to None, the HDC_EV_RECEIVE event will fire if RThreshold is 0. See the RThreshold property and the OnComm event for more information.

When AutoProcess is set to HDC_AUTOPROCESS_SERIAL, all data that is received by the serial port is automatically displayed in the terminal window (assuming that the Emulation property is non-zero).
When AutoProcess is set to HDC_AUTOPROCESS_KEY, all keys pressed in the terminal window are sent out the serial port (assuming that the Emulation property is non-zero).

When AutoProcess is set to HDC_AUTOPROCESS_BOTH, HDCTerm acts as a terminal program, both displaying data that comes in over the serial port and also sending keystrokes out the serial port.

NOTE: When AutoProcess is used, HDC_EV_RECEIVE events will not fire the OnComm event, and the receive and transmit buffers may not be directly accessed (i.e., Input, Output, and LineInput properties may not be used). AutoProcess is meant to provide a direct link between the serial port and the terminal window. If HDCTerm were to let you tap into that connection, AutoProcess would not be as fast as it is.

The best way to get around this is to temporarily disable AutoProcess when you need to directly manipulate the input and output of data, and re-enable AutoProcess when you are done.

AutoSize

Purpose: The AutoSize property indicates whether the HDCTerm control should automatically size itself so that all rows and columns are visible.

Syntax: [form.]Comm1.AutoSize[= boolean%]

Design Time Interface:True/False List Box

Data Type: Boolean Integer

Usage: Read/Write at design time and runtime.

Comments:
When using terminal emulation and AutoSize is True, the terminal window will automatically size itself to the number of rows and columns specified by the Rows and Columns properties, taking into account the size of the current font.

Note that when using a proportional font, the width of the terminal window may be less than the width of some lines of text, because HDCTerm calculates the terminal width to be the average character width of the font multiplied by the number of columns as specified by the Columns property. This makes more sense than either using the widest character in the font set as a guide, or re-sizing the width of the terminal window on the fly.

BackSpace (Custom Property)

Purpose: Toggles destructive BackSpace characters.

Syntax: [form.]Comm1.BackSpace[=integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments:
When BackSpace is set to HDC_BACKSPACE_DESTRUCTIVE (default), backspace characters are destructive, and erase the previous character in the terminal window (when Emulation is used). When BackSpace is set to HDC_BACKSPACE_NON_DESTRUCTIVE, backspace characters are non-destructive, and do not erase the previous charecter in there terminal (while Emulation is used).

CaptureFilename (Custom Property)

Purpose: When the CaptureFilename is set to a valid filename HDCTerm will initiate data capture, which saves all data that comes over the serial port to the filename specified. The way in which the data is filtered is determined by the CaptureMode property.

Syntax: [form.]Comm1.CaptureFilename[= filename$]

Design Time Interface: None

Data Type: String

Usage: Write only at runtime only.

Comments: Capture mode is a term used in terminal program software. It simply lets you save whatever data you receive to a file. The capture file is opened in shared mode, allowing your program to open it for read only. However, you can not write to the file until the file has been closed, by setting the CaptureFilename to an empty string ("").

To close the capture file, set CaptureFilename to an empty string ("").

CaptureMode (Custom Property)

Purpose: The CaptureMode property sets and returns the capture mode, which determines how data is filtered before being saved to the file specified by the CaptureFilename property.

Syntax: [form.]Comm1.CaptureMode[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments: Valid settings are:

HDC_CAPTURE_STANDARD	'-- Standard
HDC_CAPTURE_BINARY	'-- Binary
HDC_CAPTURE_VISIBLE	'-- Visible

"Capture mode" is a term used in terminal program software. It simply lets you save whatever data you receive to a file. When CaptureMode is set to HDC_CAPTURE_STANDARD, data is saved without color codes in the exact order that it was received. For example, you could receive an ANSI string to draw a line at row 20, followed by an ANSI string to draw a line at row 19. The capture file would contain row 20 followed by row 19, not the way it appears on the screen.

When CaptureMode is set to HDC_CAPTURE_BINARY, nothing is filtered and data is saved verbatim.

When CaptureMode is set to HDC_CAPTURE_VISIBLE, only the text is saved and written to disk, exactly as it looks in the terminal window (without the color codes). See the CaptureFilename property for information on capturing received data.

CDHolding (Custom Property)
Purpose: The CDHolding property returns True if the Carrier Detect (CD) line is high (if a carrier is present).

Syntax: [boolean% =][form.]Comm1.CDHolding

Design Time Interface: None

Data Type: Boolean Integer

Usage: Read only at runtime only.

Comments: When the CD line is low (CDHolding = False) and CDTimeout number of milliseconds have passed, HDCTerm sets the CommEvent property to HDC_ER_CDTO (Carrier Detect Timeout Error) and fires the OnComm event.

It is especially important to trap a loss of carrier in a host-remote application such as a bulletin board, because the caller can hang up (dropping the carrier) at any time.

ColorFilter (Custom Property)

Purpose: The ColorFilter property sets and returns the color mode of the terminal window.

Syntax: [form.]Comm1.ColorFilter[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments: Valid settings are:

HDC_COLOR_FULL	-- Full Color
HDC_COLOR_GREY	'-- Grayscale
HDC_COLOR_MONO	'-- Black and White

When ColorFilter is set to HDC_COLOR_FULL, the BackColor and ForeColor are active in full color.

When ColorFilter is set to HDC_COLOR_GREY, all color values are scaled to gray.

When ColorFilter is set to HDC_COLOR_MONO, all color values are scaled to either black or white. You would use gray scaled or monochrome colors if, for example, you are running on a laptop with 64-shade VGA video hardware.

Columns (Custom Property)

Purpose: The Columns property sets and returns the number of columns in the terminal emulator.

Syntax: [form.]Comm1.Columns[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments:
The Columns property does not refer to the width of the control. Rather, it refers to the number of fixed-pitch columns when using terminal emulation, just as a standard DOS text screen has 80 columns.
NOTE: When Emulation is set to HDC_EMULATION_NONE (iconic), no memory is used and the Columns property is ignored.

CommEvent (Custom Property)

Purpose: The CommEvent property returns the most recent communications event or error.

Syntax: [integer% =][form.]Comm1.CommEvent

Design Time Interface: None

Data Type: Integer

Usage: Read only at runtime only.

Comments:
Although the OnComm event is triggered whenever a communications error or event occurs, the CommEvent property holds the code number for that error or event. It is therefore necessary to read the CommEvent property once the OnComm event is triggered in order to determine the actual error or event.

The Code returned by CommEvent is one of the following values as specified in the file, COMMCTRL.BAS:

Communications Events:

SETTING	DESCRIPTION
HDC_EV_CD	Change in Carrier Detect line.
HDC_EV_CTS	Change in Clear To Send line.
HDC_EV_DSR	Change in Data Set Ready line.
HDC_EV_EOF	End Of File (ASCII character 26) character received.
HDC_EV_RING	Ring detected.
HDC_EV_RECEIVE	Received RThreshold number of characters.
HDC_EV_SEND	There are fewer than SThreshold number of characters in the output buffer.
HDC_EV_XFER	File transfer event. Read the XferStatus property for the current file transfer status.

Communications Errors:

SETTING	DESCRIPTION
HDC_ER_BREAK	A Break signal was received.
HDC_ER_INTO	Input Timeout. The requested number of characters specified by the InputLen property could not be returned by the Input or LineInput properties before InTimeout number of milliseconds have passed.
HDC_ER_CDTO	Carrier Detect Timeout. The CD line was low for CDTimeout number of milliseconds while trying to transmit a character. Carrier Detect is also known as the Receive Line Signal Detect (RLSD).
HDC_ER_CTSTO	Clear To Send Timeout. The CTS line was low for CTSTimeout number of milliseconds while trying to transmit a character.
HDC_ER_DSRTO	Data Set Ready Timeout. The DSR line was low for DSRTimeout number of milliseconds while trying to transmit a character.
HDC_ER_FRAME	Framing Error. The hardware detected a framing error. Generally, you cannot do anything about this error, except to try to send or
receive	again.
HDC_ER_OVERRUN	Port Overrun. A character was not read from the hardware before the next character arrived, and was thus lost. If you get this error
under	Windows 3.0, decrease the value of the Interval property. See
the	Interval property description elsewhere in this manual.
HDC_ER_RXOVER	Receive Buffer Overflow. There is no room in the input buffer.
HDC_ER_RXPARITY	Parity Error. The hardware detected a parity error.

HDC_ER_TXFULL	Transmit Buffer Full. The output buffer is full while trying to queue a character.

CommPort (Custom Property)
Purpose: The CommPort property sets and returns the communications port number.

Syntax: [form.]Comm1.CommPort[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments:
If the specified port is invalid, HDCTerm generates the Visual Basic Device Unavailable error (#68), causing an error dialog box to pop up. To prevent this from happening, you could use On Error when you open the port. See the example below. It is very important that you set the CommPort property before opening the port.

Example:

```
'-- The following example will trap an error 68 caused by opening an navailable communications
port.
      On Error Resume Next
      Do
        In$ = InputBox$("Enter Comm Port:")
        PortNum = Val(In$)
        If PortNum Then
          Comm1.CommPort = PortNum
          Comm1.PortOpen = True
          If Err Then
            MsgBox "Port is invalid, try again"
          End If
        Else
          Exit Do
        End If
      Loop
```

CTSHolding (Custom Property)

Purpose: The CTSHolding property returns True if the Clear To Send line is high.

Syntax: [boolean% =][form.]Comm1.CTSHolding

Design Time Interface: None

Data Type: Boolean Integer

Usage: Read only at runtime only.

Comments:
When the CTS line is low (CTSHolding = False) and CTSTimeout number of milliseconds have passed, HDCTerm sets the CommEvent property to HDC_ER_CTSTO (Clear To Send Timeout) and fires the OnComm event. The CTS line is used in RTS/CTS hardware handshaking. The CTSHolding property gives you a way to manually poll the CTS line. See the Handshaking property description for more information on handshaking protocols.

CTSTimeout   (Custom Property)

Purpose: The CTSTimeout property sets and returns the number of milliseconds to wait for the Clear To Send signal before setting the CommEvent property to HDC_ER_CTSTO and firing the OnComm event.

Syntax: [form.]Comm1.CTSTimeout[= long&]

Design Time Interface: Edit Window

Data Type: Long Integer

Usage: Read/Write at design time and runtime.

Comments: When the CTS line is low (CTSHolding = False) and CTSTimeout number of milliseconds have passed, HDCTerm sets the CommEvent property to HDC_ER_CTSTO (Clear To Send Timeout) and fires the OnComm event.

The CTS line is used in RTS/CTS hardware handshaking. See also the CTSHolding property which gives you a means to manually poll the CTS line. Setting CTSTimeout to 0 disables CTS Timeout detection See also the Handshaking property description for more information on handshaking protocols.

CursorRow (Custom Property)

Purpose: The CursorRow property sets and returns the current cursor row position (1 = top row).

Syntax: [integer% =][form.]Comm1.CursorRow

Design Time Interface: None

Data Type: Integer

Usage: Read/Write at runtime only.

Comments: Row coordinates range from 1 to the number of columns specified by the columns property.
NOTE: A value of 0 is treated as 1.

CursorType (Custom Property)

Purpose: The CursorType property sets and returns the type of cursor displayed.

Syntax: [form.]Comm1.CursorType[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments:

The CursorType property can have one of the following values:

HDC_CURSOR_BLOCK	'Block Cursor
HDC_CURSOR_VBAR	'Vertical Bar Cursor

The default value is HDC_CURSOR_BLOCK.

Disp (Custom Property)

Purpose: The Disp property writes the specified string to the terminal using the current emulation.

Syntax: [form.]Comm1.Disp[= string$]

Design Time Interface: None

Data Type: String

Usage: Write only at runtime only.

Comments:

NOTE: When you display a string with the Disp property, the string is only sent to the terminal window, not out the communications port. After displaying text with Disp, the cursor position is reset to the end of the displayed text. Text is always printed at the current cursor position.

It may not be obvious that a string can contain carriage returns; however, you can display more than one line of text when you use Disp by embedding Chr$(13) characters in the string.

DSRHolding (Custom Property)

Purpose: The DSRHolding property returns True if the Data Set Ready line is high.

Syntax: [boolean% =][form.]Comm1.DSRHolding

Design Time Interface: None

Data Type: Boolean Integer

Usage: Read only at runtime only.

Comments:
When the DSR line is low (DSRHolding = False) and DSRTimeout number of milliseconds have passed, HDCTerm sets the CommEvent property to HDC_ER_DSRTO (Data Set Ready Timeout) and fires the OnComm event.

This property is useful when writing a DSR/DTR handshaking routine for a DTE (Data Terminal Equipment) machine.

DSRTimeout   (Custom Property)

Purpose: The DSRTimeout sets and returns the number of milliseconds to wait for the Data Set Ready signal before setting the CommEvent property to HDC_ER_DSRTO and firing the OnComm event.

Syntax: [form.]Comm1.DSRTimeout[= long&]

Design Time Interface: Edit Window

Data Type: Long Integer

Usage: Read/Write at design time and runtime.

Comments:
When the DSR line is low (DSRHolding = False) and DSRTimeout number of milliseconds have passed, HDCTerm sets the CommEvent property to HDC_ER_DSRTO (Data Set Ready Timeout) and fires the OnComm event.

This property is useful when writing a DSR/DTR handshaking routine for a DTE (Data Terminal Equipment) machine.

Setting DSRTimeout to 0 disables DSR Timeout detection.

See also the DSRHolding property which gives you a means to manually poll the DSR line.

DTREnable (Custom Property)

Purpose: The DTREnable property toggles the state of the DTR line.

Syntax: [form.]Comm1.DTREnable[= boolean%]

Design Time Interface: True/False List Box

Data Type: Boolean Integer

Usage: Read/write at design time and runtime.

Comments:

When DTREnable is set to True the Data Terminal Ready line is set high (True) when the port is opened, and low (False) when the port is closed. When DTREnable is set to False the DTR line always remains low.

Emulation (Custom Property)

Purpose: The Emulation property specifies the current terminal emulation.

Syntax: [form.]Comm1.Emulation[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/write at design time and runtime.

Comments:
The Emulation property can have one of the following values:

HDC_EMULATION_NONE	'-- No Emulation
HDC_EMULATION_TTY	'-- TTY
HDC_EMULATION_ANSI	'-- ANSI
HDC_EMULATION_VT52	'-- VT-52
HDC_EMULATION_VT100	'-- VT100

When Emulation is set to HDC_EMULATION_NONE, the control is invisible at run-time and iconic at design-time. Although the control is invisible at runtime, it will continue to process serial communications.

FastScroll (Custom Property)

Purpose: The FastScroll property determines if the HDCTerm emulation window should perform faster block scrolls hen it falls behind by a predetermined number of characters

Syntax: [form.]Comm1.FastScroll[= boolean%]

Design Time Interface: True/False List Box

Data Type: Boolean Integer

Usage: Read/Write at design time and runtime.

Comments: When using terminal emulation and FastScroll is True, the terminal window will automatically perform multiline scrolls when certain thresholds of characters are in the receive buffer. This allows the screen to be updated faster while sacrificing a smoother scrolling display.

ForeColor (Custom Property)

Purpose: The ForeColor property sets and returns the foreground color of the terminal window when emulation is used. Note that the color values 0-15 are used. These values are the same as DOS' text-mode screen colors, and not the standard Visual Basic RGB values.

Syntax: [form.]Comm1.ForeColor[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/write at design time and runtime.

Comments: The BackColor and ForeColor properties accept values from 0 to 15 only, to maintain compatibility with terminal emulations that were designed for DOS.

InBufferCount (Custom Property)

Purpose: The InBufferCount property returns the number of characters waiting in the receive buffer.

Syntax: [form.]Comm1.InBufferCount[= integer%]
          [integer% =][form.]Comm1.InBufferCount

Design Time Interface: None

Data Type: Integer

Usage: Read/write at runtime only.

Comments:
Do not confuse this property with InBufferSize, which reflects the total size of the input buffer. InBufferCount refers to the number of   characters that have been received by the modem and are waiting in the input buffer for you to take them out.

You can flush the input buffer by setting the InBufferCount property to zero.

Example:
This example shows how to poll for received data:

```
'--- Specifies to receive all available data
Comm1.InputLen = 0
Do
  '--- Allow other processes to continue
  Dummy = DoEvents()
  '--- Check for data
  If Comm1.InBufferCount Then
    '--- Read data
    In$ = Comm1.Input
  End If
Loop
```

InBufferSize (Custom Property)

Purpose: The InBufferSize property sets and returns the size of the input buffer in bytes.

Syntax: [form.]Comm1.InBufferSize[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/write at design time and runtime.

Comments: Do not confuse this property with InBufferCount, which reflects the number of bytes currently waiting in the input buffer. InBufferSize refers to the total size of the input buffer. The default size is 2048 bytes.

NOTE: If handshaking is not being used and your receive buffer is too small, you run the risk of overflowing the buffer. As a general rule start with the default size of 2048. If you experience errors, increase the buffer size to suit your application. If you are implementing binary file transfers, you may need to increase the Input buffer size to at least 3K.

Input (Custom Property)

Purpose: The Input property returns and removes a string of characters from the input buffer.

Syntax: [string$ =][form.]Comm1.Input

Design Time Interface: None

Data Type: String

Usage: Read only at runtime only.

Comments: The number of characters that are read is determined by the InputLen property. Setting InputLen to zero tells Input to read the entire contents of the input buffer. If InTimeout number of milliseconds have passed before InputLen characters have been received, the CommEvent property is set to HDC_ER_INTO and the OnComm event is fired.

Example:
This example shows how to poll for received data:

```
'--- specify to receive all available data
Comm1.InputLen = 0
Do
  '--- Allow other processes to continue
  Dummy = DoEvents()
  '--- Check for data
  If Comm1.InBufferCount Then
    '--- Read data
    In$ = HDCTerm1.Input
  End If
Loop
```

Interval (Custom Property)

Purpose: The Interval property sets the interval in milliseconds for the control to use when checking for events under Windows 3.0.

Syntax: [form.]Comm1.Interval[= long&]

Design Time Interface: Edit Window

Data Type: Long Integer

Usage: Read/write at design time and runtime.

Comments: This property is needed only for applications that run under Windows 3.0. Under 3.0 the control has to manually poll the hardware port for data at a given interval. Under Windows 3.1, however, this is not necessary and the Interval property need not be used.

InTimeout (Custom Property)

Purpose: The InTimeout property sets and returns the number of milliseconds to wait before failing when using the Input and LineInput properties to read data from the input buffer.

Syntax: [form.]Comm1.InTimeout[= long&]

Design Time Interface: Edit Window

Data Type: Long Integer

Usage: Read/write at design time and runtime.

Comments: When Input or LineInput are used, and the required amount of data has not been received, HDCTerm will wait InTimeout number of milliseconds for data to arrive at the input buffer before setting the CommEvent property to HDC_ER_INTO and firing the OnComm event.

KeyTranslation (Custom Property)

Purpose: The KeyTranslation property determines whether HDCTerm should translate certain non-character keys and send them out of the serial port.

Syntax: [form.]Comm1.KeyTranslation[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/write at design time and runtime.

Comments: The KeyTranslation property only has an effect when AutoProcess is set to either HDC_AUTOPROCESS_KEY or HDC_AUTOPROCESS_BOTH. This property is only patialy implemented at this time, but will be expanded in the future. Valid settings are:

HDC_KEY_NONE	'Default. No Translation Done.
HDC_KEY_MANUAL	'Not currently implemented
HDC_KEY_VT100	'VT100 key translation

When KeyTranslation is set to HDC_KEY_VT100, the following keys are translated into the apropriate VT100 escape sequences.

CursorUp	(ESC) [A
CursorDown	(ESC) [B
CursorRight	(ESC) [C
CursorLeft	(ESC) [D
F1	(ESC) OP
F2	(ESC) OQ
F3	(ESC) OR
F4	(ESC) OS

Notification (Custom Property)

Purpose: Sets and returns the method by which received characters are processed internally.

Syntax: [form.]Comm1.Notification[=integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments: When Notification is set to HDC_NOTIFICATION_MANUAL, HDCTerm will use the Windows 3.0 polling method of communications. The Interval property will be functional.
When Notification is set to HDC_NOTIFICATION_DRIVER, HDCTerm will use the Windows 3.1 notification method of communications. The property will be non-functional. Windows 3.0 and Windows 3.1 use two different methods to handle communications internally, and HDCTerm for Windows supports both of them. Under Windows 3.0, and application has to poll the communications driver for received characters. In lay terms, the application has to "ASK" Windows if any characters have been received. The Windows 3.1 COMM.DRV, in contrast, used "event notification", which means that the driver "notifies" the application when x number of characters have been received. HDCTerm supports both of these methods, and automatically uses the method employed by the communications driver on the system it is running on.

The Notification property returns (at runtime) the method employed by the Windows communications driver (COMM.DRV), and allows you to change the method, in case it is necessary to do so.

For example, if you are using Windows 3.0, Notification will be set to HDC_NOTIFICATION_MANUAL at Runtime. If you try to change it to HDC_NOTIFICATION_DRIVER, HDCTerm will return "Invalid property value" because event notification is not available in Windows 3.0, only in 3.1 and higher. If you are using Windows 3.1, Notification will be set to HDC_NOTIFICATION_DRIVER at Runtime. Since using the notification method at high baud rates can sometimes produce unpredictable results, you have the option to toggle back to manual mode, in which all comm events are polled for. another way to increase communications reliability is to use the TurboComm replacement communicationsdriver, which is available from Pacific Commware.

They can be reached at (503) 482-2744.

NOTE: IF YOU ARE USING TURBOCOMM, FROM PACIFIC COMMWARE, YOU MUST SET THIS PROPERTY TO HDC_NOTIFICATION_MANUAL!

NullDiscard (Custom Property)

Purpose: When NullDiscard is set to True, Chr$(0) NULL characters are not transferred from the hardware port to the receive buffer.

Syntax: [form.]Comm1.NullDiscard[= boolean%]

Design Time Interface: True/False List Box

Data Type: Boolean Integer

Usage: Read/write at design time and runtime.

Comments: Setting NullDiscard to True effectively filters out null characters from being received. This may be useful in some situations, but is not generally necessary.

OutBufferCount (Custom Property)

Purpose: The OutBufferCount property returns the number of characters waiting in the output buffer and can optionally be used to flush the output buffer.

Syntax: [integer% =][form.]Comm1.OutBufferCount

Design Time Interface: None

Data Type: Integer

Usage: Read/write at runtime only.

Comments: Do not confuse this property with OutBufferSize, which reflects the total size of the output buffer. OutBufferCount refers to the number of characters that are waiting to be sent in the output buffer. The default output buffer size is 2048. If you are implementing file transfers, you should set the output buffer to at least 3K. You can flush the output buffer by setting the OutBufferCount property to zero.

Output (Custom Property)

Purpose: The Output property writes a string of characters to the output buffer.

Syntax: [form.]Comm1.Output[= string$]

Design Time Interface: None

Data Type: String

Usage: Write only at runtime only.

Comments: The following example shows how to dial a phone number by sending a string to the modem through the serial port:

```
DialString$ = "ATDT555-1212" + Chr$(13)
Comm1.Output = DialString$
```

ParityReplace (Custom Property)

Purpose: The ParityReplace property sets and returns the character that will replace an invalid character in the data stream when a parity error occurs.

Syntax: [form.]Comm1.ParityReplace[= string$]

Design Time Interface: Edit Window

Data Type: String

Usage: Read/write at design time and runtime.

Comments:
The parity bit refers to a bit that is transmitted along with a specified number of data bits to provide a small amount of error checking. When a parity bit is used, HDCTerm adds up all the bits that are set (having a value of 1) in the data and tests the sum as being odd or even according to the parity setting used when the port was opened.

If a parity error is detected, HDCTerm discards that character and substitutes the character you specify to indicate the error. By default the control uses a question mark (?) character for replacements.

Setting ParityReplace to an empty string () disables parity checking.When a parity error is detected, HDCTerm sets the CommEvent property to HDC_ER_RXPARITY, and fires the OnComm event.
NOTE: If you are connecting to the CompuServe Network, make sure ParityReplace is set to an empty string ().

See the Communications Tutorial in Chapter 5 of the user manual for more information on serial communications theory.

PortOpen (Custom Property)

Purpose: The PortOpen property sets and returns the state of the communications port (open or closed).

Syntax: [form.]Comm1.PortOpen[= boolean%]

Design Time Interface: None

Data Type: Boolean Integer

Usage: Read/write at runtime only.

Comments: Setting the PortOpen property to True opens the port. Setting it to False closes the port. Make sure that the CommPort property is set to a valid port number before opening the port.
Also, the Baud rate specified in the Settings property must be supported by your modem and Windows communications driver. If the Settings property contains communications settings that Windows or your serial hardware does not support, then HDCTerm will be unable to access the port. HDCTerm automatically closes the communications port when your application is terminated.

Note that when you close the port, the buffers are automatically flushed. You can, however, wait for any remaining data to be sent with the following code:

```
Do
  Dummy = DoEvents()                  'Enable control to finish sending
Loop Until Comm1.OutBufferCount = 0   ' data until the buffer is empty
Comm1.PortOpen = False                 'Then close the port.
```

Example:
The following example opens a communications port at 9600 baud, with no parity, with 8 data bits and 1 stop bit, and traps error #68 caused by opening an unavailable communications port:

```
On Error Resume Next
Comm1.Settings = "9600,N,8,1"
Do
  In$ = InputBox$("Enter Comm Port:")
  PortNum = Val(In$)
  If PortNum Then
    Comm1.CommPort = PortNum
    Comm1.PortOpen = True
    If Err Then
      MsgBox "Port is invalid, try again"
    End If
  Else
    Exit Do
  End If
Loop
```

RThreshold (Custom Property)

Purpose: The RThreshold property sets and returns the number of characters to receive before HDCTerm sets the CommEvent property to HDC_EV_RECEIVE and triggers the OnComm event.

Syntax: [form.]Comm1.RThreshold[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/write at design time and runtime.

Comments: Setting RThreshold to 1 causes HDCTerm to fire the OnComm event, every time a single character is placed in the receive buffer. Setting RThreshold to 0 disables firing the OnComm event when characters are received.

NOTE: You will normally want to set RThreshold to 1 unless you are certain that you will receive fixed-length block of data you could lose characters. For example, if you set RThreshold to 2 and you receive a string of data with an odd number of bytes, HDCTerm will not trigger the OnComm event when the last character is received because it is only one byte.

RTSEnable (Custom Property)

Purpose: The RTSEnable property toggles the state of the RTS line.

Syntax: [form.]Comm1.RTSEnable[= boolean%]

Design Time Interface: True/False List Box

Data Type: Boolean Integer

Usage: Read/write at design time and runtime.

Comments: When RTSEnable is set to True the Request To Send line is set high (True) when the port is open, and low (False) when the port is closed. When RTSEnable is set to False, the RTS line is always low (unless RTS handshaking is used).

When you use RTS/CTS handshaking you must set the RTSEnable property to TRUE, since the RTS line alternates between a high and low state when RTS/CTS handshaking is used. See the Handshaking property description for more information on handshaking protocols.

ScrollRows (Custom Property)

Purpose: The ScrollRows property sets and returns the number of rows of text to allocate to the scroll-back buffer.

Syntax: [form.]Comm1.ScrollRows[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments: When you use terminal emulation, you can optionally allocate memory for a scroll-back buffer with the ScrollRows property. The scroll-back buffer holds text that has been received, allowing the user to scroll back to view previously received text.

Settings (Custom Property)

Purpose: The Settings property sets and returns the baud rate, parity, data bits and stop bits parameters.

Syntax: [form.]Comm1.Settings[= string$]

Design Time Interface: Edit Window

Data Type: String

Usage: Read/Write at design time and runtime.

Comments: The setting string 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. Valid baud rates are determined by both the installed Windows communications driver, and the type of UART that your serial port uses.

NOTE: Some PS/2 computers have UARTs that do not support baud rates greater than 19,200. Also, the Windows 3.0 API does not allow baud rates greater than 19,200.

Valid Parity Values			
Setting	Description	Valid Data Bit Values	Valid Stop Bit Values
E	Even	4	1
N	None	5	1.5
O	Odd	6	2
S	Space	7	
M	Mark	8	

NOTE: If the Settings string is not valid when the port is opened, HDCTerm will display a Visual Basic - Invalid Property Value dialog box. You can suppress the dialog box and handle the error manually by adding the line On Error Resume Next to your code, and testing the ERR function after setting the Settings property.

Example:
The following example sets the control's port to communicate at 9600 Baud with no parity checking, 8 data bits and 1 stop bit:

```
Comm1.Settings = "9600,N,8,1"
```

SmoothScroll (Custom Property)

Purpose: The SmoothScroll property determines whether the smooth scrolling feature is used.

Syntax: [form.]Comm1.SmoothScroll[= boolean%]

Design Time Interface: True/False List Box

Data Type: Boolean Integer

Usage: Read/write at design time and runtime.

Comments:
When SmoothScroll is set to TRUE, the text in the emulator window will scroll a pixel at a time as oposed to a line at a time. This produces a very nice effect while sacrificing speed. If you use smooth scrolling it is very important that you implement some sort of hadshaking, as it is very likely you will be filling up your receive buffer.

SThreshold (Custom Property)

Purpose: The SThreshold property sets and returns the minimum number of bytes allowable in the output buffer before HDCTerm sets the CommEvent property to HDC_EV_SEND and fires the OnComm event.

Syntax: [form.]Comm1.SThreshold[= integer%]

Design Time Interface: Edit Window

Data Type: Integer

Usage: Read/write at design time and runtime.

Comments: If the number of bytes in the buffer is greater than SThreshold, and then becomes SThreshold, the CommEvent property is set to HDC_EV_SEND and the OnComm event is fired. Setting the SThreshold property to 0 disables firing of the OnComm event for data transmission events.

Upload (Custom Property)

Purpose: The Upload property initiates a binary file transfer to upload (send) a file.

Syntax: [form.]Comm1.Upload[= filename$]

Design Time Interface: None

Data Type: String

Usage: Read/Write at runtime only.

Comments: When the Upload property is set to a string that contains a valid filename, HDCTerm starts an upload using the file transfer protocol defined by the XferProtocol property. Immediately after setting the Upload property, HDCTerm returns control to your program. The transfer will happen in the background, and the status of the transfer can be monitored with the XferStatus property.
The file transfer status can be displayed with the built in file transfer dialog box. You may select the type of dialog box you want to display with the XferStatus Dialog property.

You can set the Upload property more than once to upload more than one file, no matter what file transfer protocol you use.

NOTE: When XferStatus is 0 (file is being transferred), you cannot access the CommPort, Input, Output, LineInput, Settings or PortOpen properties, or set the InBufferCount and OutBufferCount properties.
See the section entitled Transferring Files With The HDCTerm for Windows Control located elsewhere in this manual for more information on binary file transfers. Also see the Upload subroutine in CTRLDEMO.BAS, which is used in CTRLDEMO.MAK, the HDCTerm for Windows custom control demo terminal program.

We are constantly updating our products, so please read the README.WRI file for the latest information about the file transfer implementation in HDCTerm for Windows.

The following file transfer protocol constants are defined in COMM- CTRL.BAS, which should be included in your project if you are using the HDCTerm for Windows custom control:

```
HDC_XMODEM_CHECKSUM         '-- XModem-Checksum
HDC_XMODEM_CRC              '-- XModem-CRC
HDC_XMODEM_1K               '-- XModem-1K
HDC_YMODEM_BATCH            '-- YModem-Batch
HDC_YMODEM_G                '-- YModem-G
HDC_ZMODEM             '-- ZModem
HDC_COMPUSERVE_BPLUS        '-- CompuServe B+
```

Examples:
The following example will start an XModem upload using the built-in file transfer status dialog box:

```
    '-- Set the protocol to XModem-Checksum
    Comm1.XFerProtocol = HDC_XMODEM_CHECKSUM
    '-- Use the built-in dialog box to display status information
    Comm1.XferStatusDialog = HDC_XFERDIALOG_MODELESS
    '-- Start the upload
    Comm1.Upload = "FILE1.ZIP"
    '-- Control immediately comes back here after setting the Upload
    '   property, and the file transfer will happen automatically in
    '   the background.
```

The following code will start a YModem-Batch upload using the built-in file transfer status dialog box:

```
'-- Set the protocol to YModem-Batch
Comm1.XFerProtocol = HDC_YMODEM_BATCH
'-- Use the built-in dialog box to display status information
Comm1.XferStatusDialog = HDC_XFERDIALOG_MODELESS
'-- Upload two files
Comm1.Upload = "FILE1.ZIP"
Comm1.Upload = "FILE2.ZIP"
'-- Control immediately comes back here after setting the Upload
'    property, and the file transfers will happen automatically in
'    the background.
```

XferCarrierAbort (Custom Property)

Purpose: The XferCarrierAbort property toggles automatic carrier detection durring file transfers.

Syntax: [form.]Comm1.XferCarrierAbort[= boolean%]

Design Time Interface: True/False List Box

Data Type: Boolean Integer

Usage: Read/write at design time and runtime.

Comments: When XferCarrierAbort is set to TRUE, a file transfer will require the presence of CD in order to proceed. If you loose carrier in the middle of a file transfer, the transfer will correctly shut down and return control of the port back to your application. If you are using a direct connection, or something that doesn't assert the CD line, you must set this property to FALSE in order to transfer files.

XferDestFilename (Custom Property)

Purpose: Sets and returns the name of a file being transferred

Syntax: [form.]Comm1.XferDestFilename[= filename$]

Design Time Interface: None

Data Type: String

Usage: Read/write at runtime only.

Comments: XferDestFilename always returns the name of the file that will be written on the destination computer. For example, let's say we want to download a file from a BBS. On the BBS, the file is named BBS.ZIP. However, we want this file to be written to our computer as, MYFILE.ZIP. To do this, we would simply set the Download property to the filename that we want to be written to disk:

```
Comm1.Download = "MYFILE.ZIP"
```

In this case, assuming that we are using a batch protocol, such as YModem-Batch, the XferSourceFilename is BBS.ZIP, and the XferDestFilename is MYFILE.ZIP
The XferDestFilename property can only be set in the OnComm event after a HDC_XFER event occurs, when the XferStatus property is either HDC_XFER_FILE_READY or HDC_XFER_FILE_START. The following sample code shows how to set the XferDestFilename property:

```
Sub Comm1_OnComm ()
  Select Case Comm1.CommEvent
  '--Did an XFER event occur?
  Case HDC_XFER
    Select Case Comm1.XferStatus
    '--Is HDCTerm preparing to transfer the file?
    Case HDC_XFER_FILE_READY, HDC_XFER_FILE_START
      '--Rename the file
      Comm1.XferDestFileName = "NEWNAME.ZIP"
    End Select
  End Select
End Sub
```

NOTE: During an upload, there is no way to be certain that XferDestFilename is accurate, because the receiver has the ability to change the name of the file as it is being received. See the XferSourceFilename property description for more information.
Also see the section entitled "Transferring Files With The HDCTerm for Windows Control" located elsewhere in this manual for more information on binary file transfers.

XferMessage (Custom Property)

Purpose: The XferMessage property returns a status message from a file transfer in progress.

Syntax: [string$ =][form.]Comm1.XferMessage

Design Time Interface: None

Data Type: String

Usage: Read only at runtime only.

Comments: XferMessage returns the status of a file transfer in progress in the form of a string. This property may be useful if you are displaying your own custom file transfer status dialog box. See the section entitled Transferring Files With The HDCTerm for Windows Control located elsewherein this manual for more information on binary file transfers.

XferProtocol (Custom Property)

Purpose: The XferProtocol property sets and returns the transfer protocol for a file transfer.

Syntax: [form.]Comm1.XferProtocol[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments: Valid settings are:

HDC_XMODEM_CHECKSUM	'-- XModem-Checksum
HDC_XMODEM_CRC	'-- XModem-CRC
HDC_XMODEM_1K	'-- XModem-1K
HDC_YMODEM_BATCH	'-- YModem-Batch
HDC_YMODEM_G	'-- YModem-G
HDC_ZMODEM	'-- ZModem
HDC_COMPUSERVE_BPLUS	'-- CompuServe B+

NOTE: We are constantly updating our products. For the current list of supported protocols see the README.WRI file.

See the section entitled Transferring Files With The HDCTerm for Windows Control located elsewhere in this manual for more information on binary file transfers.

XferSourceFilename (Custom Property)

Purpose: Returns the original name of a file during a file transfer.

Syntax: [form.]Comm1.XferSourceFilename[= filename$]

Design Time Interface: None

Data Type: String$

Usage: Read only at runtime only.

Comments: During an upload, the XferSourceFilename property returns the original filename, or the name of the file on your computer's disk. During a download, the XferSourceFilename property returns the original name of the file on the other computer's disk, but only if the protocol being used is a batch protocol, such as YModem-Batch or ZModem. If you are using a non-batch protocol, such as XModem, the XferSourceFilename property will return an empty string ().

See the XferDestFilename property description for more information.

Also see the section entitled Transferring Files With The HDCTerm for Windows Control located elsewhere in this manual for more information on binary file transfers.

XferStatusDialog (Custom Property)

Purpose: The XferStatusDialog property sets the type of file transfer status dialog box that will be used for file transfers.

Syntax: [form.]Comm1.XferStatusDialog[= integer%]

Design Time Interface: Drop-Down List Box

Data Type: Integer

Usage: Read/Write at design time and runtime.

Comments: Valid settings are:

HDC_XFERDIALOG_NONE	'-- No Dialog Box
HDC_XFERDIALOG_MODELESS	'-- Modeless Dialog Box
HDC_XFERDIALOG_MODAL	'-- Modal Dialog Box

When XferStatusDialog is set to HDC_XFERDIALOG_NONE, no dialog box will be shown. This is useful if you want to create your own file transfer dialog box.

When XferStatusDialog is set to HDC_XFERDIALOG_MODELESS, the dialog box will be modeless (or not-modal). When XferStatusDialog is set to HDC_XFERDIALOG_MODAL, the dialog box will be modal.

See the section entitled Transferring Files With The HDCTerm for Windows Control located elsewhere in this manual for more information on binary file transfers.

XferTransferred (Custom Property)

Purpose: The XferTransferred property returns the number of bytes that have been transferred in a file transfer that is in progress.

Syntax: [long& =][form.]Comm1.XferTransferred

Design Time Interface: None

Data Type: Long Integer

Usage: Read only at runtime only.

Comments: This property may be useful if you are displaying your own custom file transfer status dialog box. See the section entitled Transferring Files With The HDCTerm for Windows Control located elsewhere in this manual for more information on binary file transfers.

OnComm (Custom Event)

Description: The OnComm event is fired whenever the value of the CommEvent property changes, indicating that either a communications event or error occurred.

Syntax: Subname_OnComm()

Comments:
The actual error or event that triggered the OnComm event is reflected in the CommEvent property. Note that setting the RThreshold or SThreshold properties to 0 disables trapping for the HDC_EV_RECEIVE and HDC_EV_SEND events respectively.

NOTE: The constants referred to below are defined in the COMM-CTRL.BAS file. You should add this module to your project to effectively use the HDCTerm control.

Example:
This example shows how to handle communications errors and events. You can insert code to handle a particular error or event after its corresponding Case statement below.

```
Sub Comm1_OnComm
Select Case Comm1.CommEvent
'-- Errors
  Case HDC_ER_BREAK                'A Break was received.
  '--- Code to handle a BREAK goes here, etc.
  Case HDC_ER_INTO                         'Input Timeout.
  Case HDC_ER_CDTO                 'CD (RLSD) Timeout.
  Case HDC_ER_CTSTO                        'CTS Timeout.
  Case HDC_ER_DSRTO                        'DSR Timeout.
  Case HDC_ER_FRAME                        'Framing Error
  Case HDC_ER_OVERRUN                      'Data Lost.
  Case HDC_ER_RXOVER                       'Input buffer overflow.
  Case HDC_ER_RXPARITY                     'Parity Error.
  Case HDC_ER_TXFULL                       'Output buffer full.
'-- Events
  Case HDC_EV_CD                           'Change in the CD line.
  Case HDC_EV_CTS                          'Change in the CTS line.
  Case HDC_EV_DSR                          'Change in the DSR line.
  Case HDC_EV_RING                         'Change in the Ring Indicator.
  Case HDC_EV_RECEIVE                      'Received RThreshold number of characters.
  Case HDC_EV_SEND                 'There are (SThreshold - 1)
                               '  number of characters in the output buffer.
  Case HDC_EV_XFER                 'File Transfer event. Check the XferStatus
                               '  property to determine the transfer event.
End Select
End Sub
```