Uso del control MSComm (Communications)

El control **Communications** permite agregar tanto una funcionalidad sencilla de comunicaciones de puerto serie, como una funcionalidad avanzada para crear una herramienta de comunicaciones completa controlada por eventos.

El control **Communications** proporciona una interfaz con un conjunto estándar de comandos de comunicaciones. Permite establecer una conexión con un puerto serie, conectar con otro dispositivo de comunicaciones (por ejemplo, un módem), emitir comandos, intercambiar datos, y supervisar y responder a varios eventos y errores que se pueden producir durante una conexión serie.

Aplicaciones posibles

- Marcar un número de teléfono.
- Supervisar la llegada de datos a un puerto serie.
- Crear un programa completo de terminal.

Fundamentos de la comunicación serie

Todos los equipos se suministran con uno o más puertos serie, que se denominan sucesivamente COM1, COM2, etc. En un equipo estándar, normalmente el *mouse* (ratón) estará conectado al puerto COM1. En el puerto COM2 puede haber conectado un módem, en COM3 un escáner, etc. Los puertos serie proporcionan un canal para la transmisión de datos desde estos dispositivos serie externos.
La función esencial del puerto serie es actuar como intérprete entre la CPU y el dispositivo serie. Al enviar datos desde la CPU a través del puerto serie, los valores de tipo **byte** se convierten en series de bits. Cuando se reciben datos, las series de bits se convierten en valores de tipo **byte**.

Para completar la transmisión de los datos es necesario otro nivel de interpretación. En el lado del sistema operativo, Windows utiliza un controlador de comunicaciones, Comm.drv, para enviar y recibir datos mediante las funciones estándar de la API de Windows. El fabricante del dispositivo serie proporciona un controlador que conecta este hardware con Windows. Cuando utiliza el control **Communications**, está ejecutando funciones de la API que interpreta el controlador Comm.drv y que se transfieren al controlador del dispositivo.

Como programador, sólo debe preocuparse por el lado de esta interacción que corresponde a Windows. Como programador en Visual Basic, sólo tiene que preocuparse de la interfaz que proporciona el control **Communications** a las funciones de la API del controlador de comunicaciones de Windows. Es decir, sólo

tiene que establecer y supervisar las propiedades y eventos del control **Communications**.

Establecer una conexión serie

El primer paso para usar el control **Communications** consiste en establecer la conexión con el puerto serie. En la tabla siguiente se enumeran las propiedades que se utilizan para esto:

Propiedades	Descripción
CommPort	Establece y devuelve el número del puerto de comunicaciones.
Settings	Establece y devuelve la velocidad en baudios, la paridad, los bits de datos y los bits de parada como una cadena.
PortOpen	Establece y devuelve el estado de un puerto de comunicaciones. También abre y cierra un puerto.

Abrir el puerto serie

Para abrir un puerto serie, utilice las propiedades **CommPort**, **PortOpen** y **Settings**. Por ejemplo:

```
' Abre el puerto serie
MSComm1.CommPort = 2
MSComm1.Settings = "9600,N,8,1"
MSComm1.PortOpen = True
```

La propiedad **CommPort** determina el puerto serie que se va a abrir. Si hay un módem conectado a COM2, en el ejemplo anterior se establece el valor a 2 (COM2) y se conecta con el módem. Puede establecer la propiedad **CommPort** a cualquier número entre 1 y 16 (el valor predeterminado es 1). Sin embargo, si establece este valor a un puerto COM que no existe en el sistema en el que se ejecuta la aplicación, se producirá un error.

La propiedad **Settings** permite especificar la velocidad en baudios, la paridad y el número de bits de datos y de parada. De forma predeterminada, la velocidad en baudios es 9600. La paridad sirve para la validación de los datos. Normalmente no se utiliza y se establece a "N". El valor de bits de datos indica el número de bits que representan un bloque de datos. El bit de parada indica cuándo se ha recibido un bloque de datos.

Después de especificar el puerto que se va a abrir y la forma en que se realizará la comunicación de los datos, para establecer la conexión puede usar la propiedad **PortOpen**, que es un valor booleano, **True** o **False**. Sin embargo, si el puerto no funciona, si la propiedad **CommPort** no se ha establecido correctamente o si el dispositivo no admite la configuración especificada, se producirá un error o puede que el dispositivo externo no funcione correctamente. Si establece la propiedad **PortOpen** a **False**, se cierra el puerto.

Trabajar con un módem

En la mayoría de los casos, usará el control **Communications** para programar la aplicación de forma que funcione con un módem. Con el control **Communications** puede usar el conjunto de comandos estándar compatible con Hayes para marcar un número de teléfono o para conectarse e interactuar con otro módem.

Una vez establecida la conexión con el puerto serie mediante las propiedades **CommPort**, **Settings** y **PortOpen**, puede usar la propiedad **Output** para activar el módem e interactuar con él. La propiedad **Output** se utiliza para emitir los comandos que controlan la interacción entre dos módem. Por ejemplo:

```
' Activa el módem y marca un número de teléfono.
MSComm1.Output = "ATDT 555-5555" & vbCr
```

En el ejemplo anterior, el comando "AT" inicia la conexión, "D" marca el número y "T" especifica que el marcado es por tonos (y no por pulsos). Debe incluir un carácter de retorno de carro (**vbCr**) cuando envíe datos a un terminal. No necesita agregar el carácter de retorno de carro cuando envíe matrices de byte.

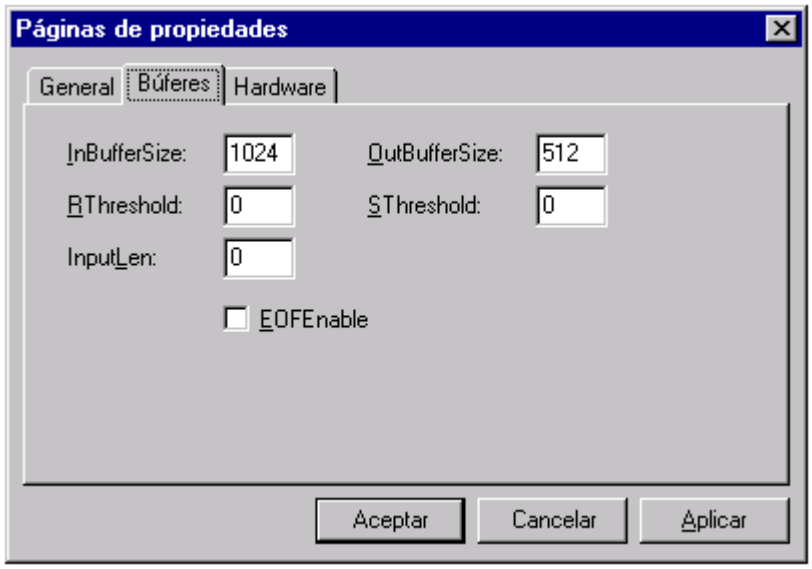
Cuando un comando se procesa con éxito, se obtendrá el código de resultado "OK". Puede comprobar este código de resultado para determinar si el comando se ha ejecutado correctamente.

Para obtener más información Si desea obtener una lista completa de los comandos compatibles con Hayes, vea la documentación del módem.

Establecer las propiedades de los búferes de transmisión y de recepción en tiempo de diseño

Cuando se abre un puerto, se crean búferes de transmisión y de recepción. Para administrarlos, el control **Communications** le proporciona diversas propiedades que puede establecer en tiempo de diseño a través de las páginas de propiedades del control.

Establecer las propiedades del búfer en tiempo de diseño



Asignación de memoria de búfer

En las propiedades **InBufferSize** y **OutBufferSize**, especifique la cantidad de memoria asignada a los búferes de recepción y transmisión. Los valores establecidos de manera predeterminada son los mostrados en la figura 2.2. Cuanto mayor sea el valor especificado, menos memoria habrá disponible para la aplicación. Sin embargo, si el búfer es demasiado pequeño, corre el riesgo de desbordarlo, a menos que utilice un protocolo.

Nota Dada la cantidad de memoria disponible en la mayoría de los equipos en la actualidad, la ubicación de la memoria de búfer tiene menor importancia porque tiene disponibles más recursos. En otras palabras, puede establecer los valores de búfer más elevados sin que afecte el rendimiento de la aplicación.

Propiedades RThreshold y Sthreshold

La propiedades Rthreshold y Sthreshold establecen o devuelven el número de caracteres recibidos en los búferes de recepción y transmisión antes de producirse el evento OnComm. El evento OnComm se utiliza para supervisar y responder a los cambios en el estado de la comunicación. Si se establecen ambas propiedades a cero (0), se impide que se produzca el evento OnComm. Si se establece un valor distinto de 0 (1, por ejemplo), el evento OnComm se producirá cada vez que se reciba un único carácter en cualquiera de los dos búferes.

Para obtener más información Si desea obtener más información acerca de propiedades, vea "Evento OnComm y propiedad CommEvent" en este mismo tema.

Propiedades InputLen y EOFEnable

Si establece la propiedad **InputLen** a 0, el control **Communications** leerá todo el contenido del búfer de recepción cuando se utilice la propiedad **Input**. Cuando lea datos de un equipo cuyo resultado tiene formato de bloques de datos de longitud fija, puede establecer el valor de esta propiedad como corresponda.

La propiedad **EOFEnable** se utiliza para indicar que se ha encontrado un carácter de final de archivo (EOF) durante la entrada de datos. Si se establece a **True**, la entrada de datos se detiene y se desencadena el evento OnComm para informarle de que se ha producido esta situación.

Para obtener más información Si desea obtener más información, vea "Administrar los búferes de recepción y transmisión" y "Evento OnComm y propiedad CommEvent", en este mismo tema.

Administrar los búferes de recepción y transmisión

Como se ha indicado anteriormente, los búferes de recepción y transmisión se crean siempre que se abre un puerto. Estos búferes se utilizan para almacenar los datos de entrada y para transmitir los datos de salida. El control **Communications** permite administrarlos a través de diversas propiedades con las que puede colocar y recuperar datos, obtener el tamaño de cada búfer y tratar datos de texto y binarios. La correcta administración de estos búferes es una parte importante del uso del control **Communications**.

Búfer de recepción

Utilice la propiedad **Input** para almacenar datos en el búfer de recepción y también para obtenerlos de él. Por ejemplo, si desea recuperar datos del búfer de recepción y mostrarlos en un cuadro de texto, puede usar código como el siguiente:

```
TxtDisplay.Text = MSComm1.Input
```

Sin embargo, para recuperar todo el contenido del búfer de recepción, primero debe establecer la propiedad **InputLen** a 0. Esto se puede realizar en tiempo de diseño o en tiempo de ejecución.

También puede recibir los datos de entrada como texto o como datos binarios, si establece la propiedad **InputMode** a una de las siguientes constantes de Visual Basic: **comInputModeText** o **comInputModeBinary**. Recuperará los datos con

formato de cadena o como datos binarios en una matriz de bytes. Utilice **comInputModeText** para los datos que empleen el juego de caracteres ANSI y **comInputModeBinary** para todos los demás datos, como los que incluyan caracteres de control incrustados, caracteres nulos, etc.

Al recibir un byte de datos, se traslada al búfer de recepción y la propiedad **InBufferCount** se incrementa en una unidad. Por tanto, es posible usar la propiedad **InBufferCount** para obtener el número de bytes del búfer de recepción. También puede borrar el contenido del búfer de recepción si establece el valor de esta propiedad a 0.

Búfer de transmisión

Utilice la propiedad **Output** para enviar comandos y datos al búfer de transmisión. Al igual que con la propiedad **Input**, puede transmitir los datos como texto o como datos binarios. Sin embargo, la propiedad **Output** debe transmitir el texto o los datos binarios mediante la especificación de una cadena o de un tipo **Variant** de matriz de bytes.

Con la propiedad **Output** puede enviar comandos, cadenas de texto o datos de una matriz de bytes. Por ejemplo:

```
' Envía un comando AT
MSComm1.Output = "ATDT 555-5555"

' Envía una cadena de texto
MsComm1.Output = " Esto es una cadena de texto"

' Envía datos de una matriz de bytes
MSComm1.Output = Out
```

Como se mencionó anteriormente, las líneas de transmisión deben terminar con un carácter de retorno de carro (**vbCr**). En el último ejemplo, Out es una variable definida como matriz de bytes: Dim Out() As Byte. Si fuera un tipo **Variant** de cadena, se definiría de este modo: Dim Out() As String.

Puede controlar el número de bytes del búfer de transmisión mediante la propiedad **OutBufferCount**. También puede borrar el contenido del búfer de transmisión si establece esta propiedad a 0.

Protocolo

Una de las tareas de la administración de los búferes de recepción y transmisión es asegurar que el tráfico de datos tenga éxito; por ejemplo, que la velocidad con la que se reciben los datos no desborde los límites del búfer.

El término protocolo hace referencia al protocolo de comunicaciones interno por el cual se transfieren los datos del puerto hardware al búfer de recepción. Cuando llega un carácter de datos al puerto serie, el dispositivo de comunicaciones tiene que llevarlo al búfer de recepción para que el programa pueda leerlo. Un protocolo de comunicaciones asegura que los datos no se pierdan por un desbordamiento del búfer, lo que puede ocurrir cuando los datos llegan al puerto tan rápido que el dispositivo de comunicaciones no puede llevarlos al búfer de recepción.

Puede establecer la propiedad **Handshaking** para especificar el protocolo de comunicación que va a usar la aplicación. De forma predeterminada, está establecida a **comNone** (ninguno). Sin embargo, puede especificar cualquiera de los protocolos siguientes:

Opción	Valor	Descripción
comNone	0	Sin protocolo (predeterminado).
comXOnXOff	1	Protocolo XOn/XOff.
comRTS	2	Protocolo RTS/CTS (Petición para emitir/Listo para emitir).
comRTSXOnXOff	3	Ambos protocolos RTS/CTS y XOn/XOff.

El protocolo que elija dependerá del dispositivo con el que se va a conectar. Si especifica **comRTSXOnXOff**, se admiten los dos protocolos.

En muchos casos, es el propio protocolo de comunicaciones quien controla el protocolo. Por lo tanto, si establece esta propiedad a algo diferente de comNone puede generar conflictos.
Nota Si establece este valor a **comRTS** o a **comRTSXOnXOff**, debe establecer la propiedad **RTSEnabled** a **True**. De lo contrario, podrá conectarse y enviar datos, pero no recibirlos.

Evento OnComm y propiedad CommEvent

Según el alcance y la funcionalidad de la aplicación, puede que necesite controlar y responder a una serie de eventos y errores que pueden producirse durante la conexión con otro dispositivo o en la recepción o transmisión de los datos.
El evento **OnComm** y la propiedad **CommEvent** le permiten interceptar y comprobar el valor de los eventos y errores de comunicación.
Cuando se produce un evento o un error de comunicación, se desencadena el evento OnComm y se modifica el valor de la propiedad **CommEvent**. De esta forma, si es necesario, puede comprobar el valor de **CommEvent** cada vez que ocurra el evento OnComm. Como las comunicaciones (especialmente a través de

líneas telefónicas) pueden ser impredecibles, la interceptación de estos eventos y errores permite una respuesta adecuada.

En la tabla siguiente se enumeran los eventos de comunicaciones que desencadenará el evento OnComm. Los valores indicados se escriben en la propiedad **CommEvent**.

Constante	Valor	Descripción
comEvSend	1	En el búfer de transmisión hay menos caracteres que los indicados en **SThreshold**.
comEvReceive	2	Se ha recibido el número de caracteres indicado en **RThreshold**. Este evento se genera continuamente hasta que utilice la propiedad **Input** para quitar los datos del búfer de recepción.
comEvCTS	3	Cambio en la línea Listo para emitir (CTS).
comEvDSR	4	Cambio en la línea Terminal de datos preparado (DSR). Este evento sólo se produce cuando la línea DSR pasa de 1 a 0.
comEvCD	5	Cambio en la línea Detector de portadora (CD).
comEvRing	6	Detectada la llamada. Puede que algunos UART (transmisores-receptores universales asíncronos) no admitan este evento.
comEvEOF	7	Recibido carácter de fin de archivo (carácter ASCII 26).

El evento OnComm también se desencadena y se escribe un valor en la propiedad **CommEvent**, cuando ocurren los errores siguientes.

Constante	Valor	Descripción
comEventBreak	1001	Se ha recibido una señal de interrupción.
comEventFrame	1004	Error de trama. El hardware ha detectado un error de trama.
comEventOverrun	1006	Puerto desbordado. No se leyó un carácter desde el hardware antes de llegar otro carácter y el primero se ha perdido.

comEventRxOver	1008	Desbordamiento del búfer de recepción. No hay espacio suficiente en el búfer de recepción.
comEventRxParity	1009	Error de paridad. El hardware ha detectado un error de paridad.
comEventTxFull	1010	Búfer de transmisión lleno. Se ha intentado colocar un carácter más en la cola mientras el búfer de transmisión estaba lleno.
comEventDCB	1011	Error inesperado al obtener el bloque de control de dispositivo (DCB) para el puerto.