Manejo de Puertos - el control Microsoft COM

El control *MSComm* proporciona comunicaciones serie para que su aplicación pueda transmitir y recibir datos a través de un puerto serie.

El control *MSComm* no está normalmente en la caja de herramientas, por lo que será necesario introducirlo mediante el menú Herramientas, Componentes.

En el formulario solamente se le ve en tiempo de diseño. El icono que lo representa en la caja de herramientas es el siguiente: 

El control *MSComm* proporciona dos formas diferentes de tratamiento de las comunicaciones:

- Las comunicaciones controladas por eventos son un método muy poderoso para el tratamiento de interacciones con el puerto serie. En muchas situaciones deseará que se le notifique cuándo tiene lugar un evento; por ejemplo, cuándo llega un carácter o cuándo se produce un cambio en las líneas de Detección de portadora (CD) o Petición de envío (RTS). En tales casos se utiliza el evento *OnComm* del control *MSComm* para interceptar y tratar estos eventos de comunicaciones. El evento *OnComm* también detecta y trata los errores en las comunicaciones. En la propiedad *CommEvent* puede ver una lista completa de todos los eventos y errores posibles en las comunicaciones.
- También puede sondear los eventos y errores si comprueba el valor de la propiedad *CommEvent* después de cada función crítica de su programa. Esta alternativa es preferible si la aplicación es pequeña y autónoma. Por ejemplo, si está escribiendo un marcador telefónico sencillo, no tiene sentido generar un evento después de recibir cada carácter, ya que los únicos caracteres que piensa recibir son las respuestas de aceptación que envía el módem.

Cada uno de los controles *MSComm* que use corresponde a un puerto serie. Si necesita tener acceso a más de un puerto serie en su aplicación, debe usar más de un control *MSComm*. La dirección del puerto y la dirección de la interrupción pueden cambiarse desde el Panel de control de Windows.

Al tratarse de un control personalizado, presenta dos formas de ver las propiedades. Si hacemos clic con el botón derecho del ratón sobre el control y vamos a propiedades, nos presenta tres cuadros de configuración de los típicos de los controles personalizados.

Si seleccionamos el control *MSComm* y pulsamos F4 , aparecerá la caja de propiedades típica de los controles de Visual Basic.

Aunque el control *MSComm* tiene muchas propiedades importantes, hay algunas con las que debe familiarizarse primero. Estas son: *CommPort, Settings, PortOpen, Input* y *Output*, las cuales se detallan a continuación junto con las restantes propiedades.

Propiedades

Existen propiedades que pueden establecerse en tiempo de diseño o en tiempo de ejecución, y otras que solamente se pueden ejecutar o consultar en solamente en tiempo de ejecución. Se

detallan a continuación las primeras. Las segundas se enumerarán tras estas, aunque se nombran algunas de estas últimas al explicar cada una de las propiedades del primer tipo.

CommPort

Indica el número del puerto serie usado. Admite los valores de 1 a 255. Cambiando esa propiedad podemos cambiar el puerto de comunicación que vamos a usar (Un PC tiene normalmente 2 puertos serie: El Com1 y el Com2). Puede tener sin grandes problemas Hardware con hasta 4 (Com3 y Com4). Si le damos a ese valor un número de puerto inexistente, dará error.

Settings

Su sintaxis es: Velocidad, Paridad, Bits de información, Bits parada

Indica la velocidad, paridad, número de bits y bits de stop (parada) que se van a usar en la comunicación.

Los valores posibles para *velocidad* (en baudios) son 50, 100, 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200 y 28800

Los valores posibles para *paridad* son :

N: No envía bit de paridad ni hace comprobación de paridad en la recepción.

O: Envía y comprueba paridad, con el criterio de paridad IMPAR

E: Envía y comprueba paridad, con criterio de paridad PAR

Los valores para el parámetro *Bits de Información* pueden ser :

7: Se envían / reciben 7 bits por trama de información.

8: Se envían / reciben 8 bits por trama de información

5: Se envían / reciben 5 bits por trama de información. Este valor de 5 bits es el típico del sistema Baudot para transmisión telegráfica (Teletipos) que se ha conservado en las comunicaciones informáticas por pura tradición. Si se elige 5 bits, los bits de parada se ponen automáticamente a 1,5 (Típico también del sistema Baudot.)

Los valores para el parámetro *Bits de parada* pueden ser :

1: Se envía un bit de parada

2: Se envían 2 bits de parada

No es posible programar 1,5 bits de parada. Sólo lo hace cuando se programan 5 bits de información y lo hace automáticamente.

Handshaking

Especifica el método de control sobre el flujo de información. En una comunicación serie se necesita conocer si el puerto puede enviar información (necesita saber si el módem está preparado para recibirla) y necesita indicarle al módem que él está preparado para recibir

información. A este proceso se le denomina *Handshaking*. (Control de Flujo)

El Control de Flujo puede hacerse de dos formas: Una mediante las señales auxiliares del puerto (RTS, CTS, DSR, DTR), que son cables adicionales que tendrán una tensión positiva respecto a los 0V del equipo si esa señal está activada, o una tensión negativa si no lo está. Este tipo de control del flujo de información es el típico para comunicarse el ordenador con un módem.

Existe otra forma de controlar el flujo de información: mediante señales especiales que se envían por los dos cables que transportan la información. Mediante estas dos señales podemos controlar que el ordenador envíe información o deje de enviarla. De igual forma, podemos indicarle al módem que envíe o no envíe. Estas señales especiales se denominan X-ON y X-OFF.

La propiedad *Handshaking* controla la forma de realizar este proceso. Puede tomar los siguientes valores :

0: No existe Control de Flujo

1: Control de Flujo mediante XON - XOFF

2: Control de Flujo mediante Request To Send (RTS) y Clear To Send (CTS)

3: Control de Flujo mediante XON - XOFF y RTS - CTS

Los tres tipos de Control de Flujo tiene cada uno su aplicación.

InBufferSize

Mediante esta propiedad establecemos el tamaño del Buffer (almacén de datos) de entrada. Este Buffer sirve para poder recibir datos sin que tenga que intervenir la aplicación continuamente para controlar el puerto de entrada.

Puede conocerse el número de caracteres presentes en el Buffer de entrada consultando el valor de la propiedad *InBufferCount*.

OutBufferSize

Mediante esta propiedad controlamos el tamaño del Buffer de salida.

El tamaño de los Buffers dependerá de la aplicación y de la velocidad de comunicación. Si la aplicación tiene muchas cosas que hacer, aparte de controlar el puerto de comunicaciones, se deberá poner un Buffer grande. Tanto mas grande cuanta mayor sea la velocidad de transferencia de datos.

Puede conocerse el número de caracteres presentes en el Buffer de salida (los que aún están por transmitir), consultando el valor de la propiedad *OutBufferCount*.

RThreshold, SThreshold

Estas dos propiedades especifican el número de caracteres que deben estar presentes en los Buffers de Recepción y Transmisión respectivamente, para que se produzca el evento *OnComm* relativo a recepción y transmisión de caracteres. (Eventos *EvReceive* y *EvSend*) Si el valor de una de estas propiedades está a 0, no se produce el evento *OnComm* correspondiente.

El valor que se debe dar a estas dos propiedades depende de la aplicación y del tiempo que queramos que la aplicación está atendiendo al puerto de comunicaciones. Concretamente para la propiedad *RThreshold* debemos pensar muy bien el valor que se le pone. Si ponemos un valor corto (1 es el mínimo), cada vez que reciba un carácter se producirá el evento *OnComm*. (Vea la descripción de eventos mas adelante). Al producirse este evento, ejecutará el procedimiento asociado a él, lo que hará perder tiempo a la aplicación, impidiéndole realizar otras funciones. Si se pone un valor muy alto, el puerto no avisará que tiene caracteres recibidos hasta que reciba un número igual al programado en esta propiedad, por lo que no podremos procesar los datos recibidos hasta que el buffer tenga ese número de caracteres en su interior. En número adecuado dependerá del tipo de aplicación que vayamos a realizar. En cualquier caso, este número será inferior al número programado para la longitud del buffer, (*InBufferSize***)**

InputLen

Por defecto, cuando se lee el Buffer de recepción, se leen todos los caracteres, quedando el Buffer vacío. Si se le asigna a esta propiedad un valor distinto de 0, cada vez que leamos el Buffer de recepción leerá un número de caracteres igual a esa cantidad, permaneciendo los caracteres restantes en el Buffer a la espera de una nueva lectura. Asignándole el valor 0 (Valor por defecto), el buffer se lee completo.

ParityReplace

Si la comunicación se realiza con bit de paridad (Par o Impar), en recepción se comprueba byte a byte la recepción de la paridad correcta. Si se recibe un Byte que no tiene paridad correcta, lo más probable es que ese Byte (carácter) se haya recibido defectuoso. Esta propiedad nos permite sustituir un carácter que ha llegado con bit de paridad incorrecto por otro carácter. (? predeterminado). Se puede sustituir por una cadena de caracteres (Error, por ejemplo).

NullDiscard

Cuando se recibe el carácter nulo (00000000) puede ser que no sirva para nada a efectos de nuestra aplicación, o que este carácter sea un dato mas. Esta propiedad acepta los valores True / False. Si es True se desprecia el carácter Nulo. Si es False, se toma como un carácter mas.

CTSTimeout

Es el tiempo (en milisegundos) que permanece esperando la señal *CTS* (Señal CTS - Dispuesto para enviar), señal de entrada al ordenador que debe estar presente antes de que el puerto comience a enviar información. El tiempo se mide desde que se pone activa la señal de salida *RTS* (Petición de envío). Si se supera este tiempo entre el instante de activación de la señal *RTS* y la recepción de la señal *CTS*, se produce el evento *CTSTO*. Poniendo 0 en esta propiedad, se deshabilita, y en estas condiciones no se producirá nunca el evento *CTSTO*.

CDTimeout

Es el tiempo máximo de espera (en milisegundos) desde que se activa la señal *DTR* hasta que se

recibe la señal *CD* (Carrier Detect - Detección de portadora). Este tiempo solamente tendrá importancia en ciertas aplicaciones donde se espere recibir *CD* continuamente. No tendrá sentido cuando la aplicación se queda en espera a recibir una comunicación, pero sin saber cuando la tiene que recibir. Si transcurre el tiempo programado en esta propiedad, ocurrirá el evento *CDTO*. Poniendo el valor 0 se deshabilita esta propiedad y no se producirá nunca el evento *CDTO*.

DSRTimeout

Similar a la anterior, pero en vez de esperar la señal *CD* se espera la señal *DSR*. Esta propiedad sí tiene sentido, ya que si, por ejemplo, estamos conectados con un módem, y nuestra aplicación se pone a la espera de recibir alguna llamada, activa la salida *DTR*, y espera recibir inmediatamente la respuesta de que el módem está dispuesto, mediante la línea *DSR*. Si transcurre el tiempo programado sin recibir la señal **DSR** se producirá el evento *DSRTO* . Poniéndola a 0, se deshabilita esta propiedad y nunca ocurrirá el evento *DSRTO***.**

RTSEnable

Activa (Pone a 1) la señal *RTS* (Request To Send - Petición de envío) Esta señal debe ponerse a 1 para indicar al módem (o al equipo que va a recibir nuestra comunicación) que deseamos enviar datos. Debe estar activada durante toda la transmisión de datos.

Cuando se pone la propiedad *Handshaking* a 2 (control con RTS / CTS) ó 3 (Control con RTS / CTS y con X-ON / X-OFF) no debemos preocuparnos de poner a 1 la señal *RTS*, pues lo hace automáticamente el puerto de comunicaciones. Esta propiedad está ahí para aplicaciones donde no se emplee ese tipo de *Handshaking* y necesitemos activar algo antes de transmitir. (Caso por ejemplo de transmisión de datos por radio, donde podemos usar esta señal de salida para activar el PTT (Push To Talk - Pulse para hablar) y poner el transmisor en marcha)

DTREnable

Activa (Pone a 1) la salida DTR (Data Terminal Ready - Terminal de Datos Listo). Esta señal se emplea para decirle al módem que el terminal (Ordenador) está preparado para recibir datos.

Se hace la misma observación que para la propiedad anterior respecto a los valores de la propiedad *Handshaking*

Interval

Indica el tiempo (en milisegundos) del intervalo entre una y otra comprobación del estado de recepción del puerto. El valor mínimo es de 55 ms.

El análisis del puerto de comunicación no tiene nada que ver con la generación del evento *OnComm*. Este evento se producirá cuando se cumplan las condiciones para ello, independientemente del tiempo programado en esta propiedad. La comprobación del puerto cada intervalo de tiempo marcado por esta propiedad solamente afecta a averiguar el estado de las líneas auxiliares *CD*, *DSR* y *CTS*, y para saber el número de caracteres existentes en los Buffers de transmisión y recepción.

Las siguientes propiedades no difieren en nada respecto a otros controles: *Left, Name, Index, Top, Tag*

Propiedades propias del tiempo de ejecución

PortOpen

Abre el puerto de comunicación. Puede tener los valores *True* (Para abrirlo) y *False* (Para cerrarlo). Si tenemos un *MSComm* con Nombre (Name) MSComm1, para abrirlo ejecutaremos la siguiente sentencia :

```
MSComm1.PortOpen = True
```

Para cerrarlo, ejecutaremos :

```
MSComm1.PortOpen = False
```

InBufferCount

Nos permite averiguar cuantos caracteres tenemos en el Buffer de entrada. Con el mismo *MSComm* anterior, comprobaremos el número de caracteres sin leer con la sentencia:

```
caracteressinleer = MSComm1.InBufferCount
```

OutBufferCount

Nos permite conocer cuantos caracteres quedan por transmitir en el Buffer de salida. Emplearemos la sentencia :

```
caracteressinenviar = MSComm1.OutBufferCount
```

Output

Envía caracteres al Buffer de salida. Debe existir un signo igual (=) entre *Output* y lo que se envía al Buffer. Para enviar la frase Curso de Visual Basic ejecutaremos la sentencia :

```
MSComm1.Output = "Curso de Visual Basic"
```

Si deseamos enviar el contenido de una variable

```
MSComm1.Output = variable
```

Input

Lee el Buffer de recepción. El número de caracteres leídos dependerá del valor de la propiedad *InputLen*. Cuando la propiedad *InputLen* tiene el valor 0, el Buffer se lee completo. Si *InputLen* tiene un valor distinto de 0, se leerá un número de caracteres igual al valor de esta propiedad.

CommEvent

Devuelve el evento mas reciente que ha ocurrido para generar el evento general *OnComm* (Vea más adelante). Esta propiedad no está disponible en tiempo de diseño y es de sólo lectura en tiempo de ejecución.

Su sintaxis es:

NombredelMSComm.CommEvent

Break

Devuelve un valor (True / False) que indica que se ha recibido la señal *Break*. Su sintaxis es:

variable = MSComm1.Break

CDHolding

Devuelve el estado de la línea de control *CD* (Detección de Portadora) Si es *True*, esa entrada está activada, si es *False*, la entrada está desactivada.

variable = MSComm1.CDHolding

CTSHolding

Devuelve el estado de la línea de control *CTS* (Dispuesto para enviar) Si es *True*, esa entrada está activada, si es *False*, la entrada está desactivada.

variable = MSComm1.CTSHolding

DSRHolding

Devuelve el estado de la línea de control *DSR* (Data Set Ready) Si es *True*, esa entrada está activada, si es *False*, la entrada está desactivada.

variable = MSComm1.DSRHolding

Eventos del MSComm

El *MSComm* tiene varios eventos, pero un solo Procedimiento: el Procedimiento *OnComm*. Este procedimiento se ejecuta cada vez que se produce alguno de los eventos del *MSComm*.

Esto quiere decir que para escribir el código apropiado en el procedimiento del *MSComm* será necesario analizar qué evento se ha producido y colocar, mediante una sentencia If .. Then el código apropiado para cada uno de los eventos que se produzcan.

Para averiguar qué evento se ha producido puede hacerse consultando el valor de la propiedad *CommEvent*.

```
If CommEvent = ComEvRing Then
    'Se ha consultado si el evento particular que ha producido el 'evento general OnComm
    'ha sido el ComEvRing (Se está recibiendo la llamada del 'teléfono). En esta sentencia
    'If  Then deberemos colocar el código necesario para que la 'aplicación se prepare para
    'recibir una comunicación a través del módem.
End If
```

Los eventos del Comm pueden identificarse por una constante o un número. La constante, como todas las de Visual Basic, tiene una expresión bastante difícil. Se pone entre paréntesis el número que identifica a ese evento. Este número debe declararse como Integer.

Se ejecutará el Procedimiento *OnComm* cuando ocurra alguno de los siguientes eventos :

ComEvCD (5)	Cambio en la línea CD. Para conocer el estado actual de esa línea (Activado/Desactivado) deberemos invocar la propiedad CDHolding.
ComEvCTS (3)	Cambio en la línea CTS. Igual que la anterior, este evento solamente nos indica que ha existido un cambio. Para averiguar el estado en que se encuentra esta línea, debemos invocar la propiedad CTSHolding.
ComEvDSR (4)	Cambio en la línea DSR. Igual que las anteriores. Debemos invocar la propiedad DSRHolding para averiguar su estado actual.
ComEvRing (6)	Cambio en la línea de detección de llamada (Ring). Este evento se produce cuando hay un cambio en la línea Ring (Detección de llamada en el módem). No existe una propiedad para averiguar el estado de la línea Ring pues no es necesario. Lo importante de esta línea es que está cambiando, es decir, el teléfono está sonando y poco importa que analicemos si la línea Ring está a 1 o a 0, pues toda llamada telefónica es intermitente. Dependiendo de la UART de su PC, puede que este evento no lo soporte.
ComEvReceive (2)	Cuando se recibe un número igual o mayor de caracteres que el indicado en la Propiedad Rthreshold.
ComEvSend (1)	Cuando quedan en el búfer de transmisión menos caracteres que los indicados en la Propiedad Sthreshold.
ComEvEOF (7)	Recibido un carácter de fin de archivo (carácter ASCII 26).
comEventBreak (1001)	Se ha recibido una señal de interrupción. (Break)
ComEventCDTO (1007)	Tiempo de espera de Detección de portadora. La línea Detección de portadora (CD) estuvo baja durante el periodo de tiempo especificado en la Propiedad CDTimeout, mientras se intentaba transmitir un

	carácter.
ComEventCTST (1002)	Tiempo de espera de Preparado para enviar. La línea Preparado para enviar (CTS) estuvo baja durante el periodo de tiempo especificado en la propiedad CTSTimeout mientras se intentaba transmitir un carácter.
ComEventDSRTO (1003)	Tiempo de espera de Equipo de datos preparado. La línea Equipo de datos preparado (DSR) estuvo baja durante el periodo de tiempo especificado en la Propiedad DSRTimeout mientras se intentaba transmitir un carácter.
ComEventOverrun (1006)	Se sobrepasó la capacidad del Buffer de entrada sin haber leído todos los caracteres. Los caracteres no leídos se han perdido. Debemos aprovechar este evento para solicitar al colateral una repetición de los datos perdidos.
ComEventRxOver (1008)	Desbordamiento del búfer de recepción. No hay espacio para más datos en el búfer de recepción.
ComEventRxParity (1009)	Error de paridad. El hardware ha detectado un error de paridad.
ComEventTxFull (1010)	Búfer de transmisión lleno. El búfer de transmisión estaba lleno cuando se ha intentado agregar un carácter a la cola de transmisión. Este error es fácil de evitar, analizando el valor de la propiedad OutBufferCount antes de enviar mas datos al buffer de salida.
ComEventDCB (1011)	Error inesperado al recuperar el Bloque de control de dispositivos (DCB) para el puerto.
ComEventFrame (1004)	Error de trama. El hardware ha detectado un error de trama.

Desarrollaremos ahora la aplicación que simule un marcador telefónico. A continuación se muestra la interfaz a utilizar:

El formulario consta de un combo llamado *cboPuertos* que muestra los puertos por los cuales podrá efectuar la comunicación. Este combo se llenará desde el evento Load del formulario mediante el método *Additem*.

Además posee una caja de texto (*txtNumero*) en la cual se podrá ingresar el número telefónico al cual desee establecer la comunicación o también podrá pulsar las teclas correspondientes y el número se mostrará en dicha caja de texto.

Las teclas mostradas conforman una matriz de controles que comparten el nombre común *cmdTeclas*. Con el fin de que se muestre en la caja de texto el número correspondiente a la tecla presionada, se define el siguiente código en el evento clic de la matriz.

```
Private Sub cmdTeclas_Click(Index As Integer)
   txtNumero.Text = txtNumero.Text & cmdTeclas(Index).Caption
End Sub
```

En el evento clic del botón de pulsación Marcar se utilizarán algunas de las propiedades de objeto MSCOMM expuestas anteriormente:

```
Private Sub cmdMarcar_Click()
   If txtNumero.Text = "" Then
      MsgBox "Debe ingresar un número.", vbCritical, "Atención"
      Exit Sub
   End If
   On Error GoTo Problema
   MSComm1.CommPort = Right(cboPuertos.Text, 1)
   MSComm1.Settings = "14400,N,8,1"
   MSComm1.PortOpen = True
   MSComm1.Output = "ATDT" & txtNumero.Text & Chr(13)
   Exit Sub
Problema:
   MsgBox Err.Description, vbCritical, "Atención"
End Sub
```

Para comenzar se verifica que el usuario haya ingresado el número de teléfono al cual se desea comunicar. Luego se establece el número del puerto de comunicaciones (mediante la propiedad *CommPort*) el cual coincide con el seleccionado por el usuario desde el combo *cboPuertos*. La sentencia siguiente *Setting* nos indica que se conectará a 14400 baudios, sin paridad, con 8 bits de datos y uno de parada. Luego se abre el puerto de comunicaciones (*PortOpen = True*). Y por último se escribe una cadena de datos en el búfer de transmisión, lo que indica que se llamará al número de teléfono ingresado en la caja de texto.

Como podrá observar se a introducido un manejador de errores para el caso de que surja un problema inesperado y produzca un error en tiempo de ejecución. Por ejemplo, el control *MSComm* genera el error 68 (El dispositivo no está disponible) si no existe el puerto cuando

intenta abrirlo con la propiedad *PortOpen*.

Para poder cortar la comunicación, en el evento clic del botón de pulsación *Colgar*, escriba lo siguiente.

```
Private Sub cmdColgar_Click()
   MSComm1.PortOpen = False
End Sub
```

Con esta sentencia se cerrará el puerto de comunicaciones.

Por último, el botón de pulsación *Salir* cerrará la aplicación mediante la sentencia *End*.

Ejecute la aplicación llamando a un teléfono que posea cerca y observe los resultados.