

# Soluciones de comunicación industrial para Windows

## Manual del Driver XFINS

### *Omron SYSMAC Host Link FINS Protocol Driver*



- El driver XFINS permite crear poderosas aplicaciones de supervisión, control y comando de sus equipos.
- El manejador de licencias le permite distribuir ilimitadamente sus aplicaciones a otras computadoras o clientes.
- Las aplicaciones se arman con VB6 ó VisualStudio.NET utilizando componentes dll/.NET o bien OCX/ActiveX/COM.
- Soporta todos los sistemas operativos Windows de 32 bits y de 64 bits, corriendo de manera nativa.

El driver XFINS forma parte de un conjunto de más de 125 drivers para comunicación serie y tcp/ip desarrollados por CPKSoft Ingeniería, empresa argentina con presencia y experiencia en el sector de comunicaciones industriales desde el año 1990.

Al igual que todos los demás drivers de CPKSoft Ingeniería, el driver XFINS se maneja a través de un componente nativo para .NET llamado NetTalk o bien con un ActiveX/OCX OLE/COM llamado HMITalk. Ambos son totalmente abiertos y se distribuyen junto con cada driver. Estos componentes multi-driver ofrecen una interfaz única para manejar cualquier driver de manera idéntica desde aplicaciones escritas con cualquiera de los lenguajes soportados por Microsoft Visual Studio.NET 2008/2010, como Visual Basic y Visual C# o bien desde Visual Basic 6.0 de 32 bits (VB6).

Cuando Ud. adquiere una licencia de nuestro driver XFINS, recibe un "número de licencia ilimitada" y un utilitario manejador de licencias que le va a permitir crear licencias runtime para sub-licenciar el driver a sus propios clientes, tantos como necesite y a cualquier número de computadoras de destino, dentro y fuera de su empresa. Las licencias ilimitadas no tienen ningún límite en la cantidad de licencias runtime que se pueden crear a partir de ellas, ni en la cantidad de equipos con los que se podrán comunicar sus aplicaciones, ni en la cantidad de puntos o tags que podrán ser accedidos desde su driver, ni ninguna otra limitación característica de otros productos similares. Es un producto ideal tanto para integradores como para usuarios finales.

**CPKSoft Ingeniería**  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012



**CPKSoft Ingeniería**  
Drivers para comunicación industrial.

# Soluciones de comunicación industrial para Windows

## Contenido

Instalación del driver .....	4
Prueba de funcionamiento del driver .....	6
Uso del componente NetTalk .....	9
La interfaz de NetTalk .....	9
Crear una aplicación .....	9
Inicializaciones típicas .....	11
Manejo de la conexión .....	12
Conexión serie .....	12
Conexión tcp/ip .....	13
Código de conexión típico .....	13
Cierre de la conexión .....	14
Manejo de lecturas .....	14
Lecturas con espera .....	15
Lecturas por eventos .....	17
Manejo de escrituras .....	18
Interfaz COM del componente NetTalk .....	18
Guía rápida de propiedades, métodos y eventos .....	21
Propiedades .....	21
Métodos .....	24
Eventos .....	26
Uso del componente HMITalk .....	28
La interfaz de HMITalk .....	28
Registro de HMITalk .....	28
Prueba de funcionamiento .....	31
Crear una aplicación .....	32
Configuración del puerto de comunicaciones .....	33
Manejo de lecturas .....	34
Lecturas automáticas por eventos .....	35
Lecturas a demanda por eventos .....	37
Lecturas con espera .....	37
Manejo de escrituras .....	38
Guía rápida de propiedades, métodos y eventos .....	39
Propiedades .....	39
Métodos .....	40
Eventos .....	41
Uso del componente LineChart .....	43
La interfaz de LineChart .....	43
Dibujar una pluma .....	43

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

Manejo de la grilla de fondo .....	45
Bandas de color .....	46
Guía rápida de propiedades, métodos y eventos .....	47
Propiedades .....	47
Métodos .....	47
Eventos .....	47
<b>Distribución de las aplicaciones .....</b>	<b>48</b>
Archivos a ser distribuidos .....	48
Generación de licencias runtime .....	49
Explicación del proceso .....	49
Obtención del número de licencia ilimitada .....	49
Obtención del PCId mediante el utilitario GetPCId.exe .....	50
Obtener PCId por código desde la aplicación .....	50
Creación del archivo .lic .....	50
<b>Especificaciones técnicas del driver XFINS .....</b>	<b>52</b>
Información general .....	52
Listado de comandos .....	52
Memory Area Read .....	52
Memory Area Write .....	54
Mensajes de error .....	56
Equipos soportados .....	58

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

## Instalación del driver

Los drivers de CPKSoft Ingeniería se distribuyen únicamente como archivos .zip o como archivos .rename-to-zip. Cuando se trate de archivos con extensión .rename-to-zip, éstos deberán ser previamente renombrados como .zip antes de ser abiertos.

El proceso de instalación de un driver consiste en copiar el contenido del archivo comprimido en una carpeta en el disco rígido y no requiere de la ejecución de ningún programa.

NetTalk.dll y todos los programas ejecutables que forman parte del entregable del driver requieren del FrameWork.Net 2.0 preinstalado en su máquina. Por su parte, los componentes ActiveX/OCX HMITalk y LineChart no requieren de ningún framework de .NET preinstalado para su uso.

Los archivos más importantes que se incluyen son:

- **XFINS.\*.tlk**: es el archivo que contiene al driver propiamente dicho y es quien tiene la inteligencia para procesar los telegramas que se envían y reciben de los equipos de acuerdo al protocolo de comunicaciones que se debe manejar. Este archivo debe ser siempre distribuido con las aplicaciones. El archivo **XFINS.32.tlk** corresponde a la versión para plataformas de destino de 32 bits y el archivo **XFINS.64.tlk** corresponde a la versión para plataformas de destino de 64 bits.
- **XFINS.Castellano.pdf**: es el manual del driver en español.
- **XFINS.English.pdf**: es el manual del driver en inglés.
- **DriverTest.exe**: es un utilitario para ensayar los comandos del driver. Puede ser ejecutado tanto en 32 bits como en 64 bits indistintamente.
- **NetTalk.dll**: es el archivo del ensamblado que corresponde al componente NetTalk, quien actúa como interfaz para manejar el driver XFINS.\*.tlk desde aplicaciones .NET. Este archivo debe distribuido con las aplicaciones cuando ha sido utilizado en su construcción. Puede ser ejecutado tanto en 32 bits como en 64 bits indistintamente. Expone además una interfaz tipo COM para poder ser utilizado desde entornos de desarrollo que lo soportan, como es el caso de VB6, PHP, etc.
- **RegNetTalk.exe**: utilitario para registrar NetTalk.dll en la GAC cuando se lo quiera utilizar como un objeto del tipo COM. Se ejecuta automáticamente en modo administrador.
- **HMITK32.ocx/HMITK64.ocx**: son los archivos que corresponden al componente ActiveX HMITalk, quien actúa como interfaz para manejar el driver XFINS.\*.tlk desde aplicaciones VB6 o .NET, o cualquier otro entorno que soporte interfaz OLE/COM. Estos archivos deben ser distribuido con las aplicaciones cuando han sido utilizados en su construcción. La versión HMITK32.ocx se utiliza para aplicaciones con plataforma de destino de 32 bits y HMITK64.ocx se utiliza para aplicaciones con plataforma de destino de 64 bits. Si el entorno de desarrollo trabaja en 32 bits, ambas versiones se utilizan (HMITK32.ocx durante diseño y HMITK64.ocx en runtime), por lo que se recomienda que ambas versiones sean registradas en la máquina de trabajo.
- **GRID32.ocx**: archivo de la grilla de apoyo para las páginas de propiedades de HMITK32.ocx. Está disponible únicamente en 32 bits, aunque esto no representa un inconveniente para crear aplicaciones de 64 bits ya que sólo se invoca en tiempo de diseño, mientras se está trabajando con las herramientas de desarrollo (VB6, VS 2008/2010 Express, etc.), quienes por lo general también trabajan únicamente en 32 bits.
- **LCHART32.ocx/ LCHART64.ocx**: son los archivos que corresponden al componente ActiveX LineChart, que se distribuye para brindar compatibilidad con aplicaciones que utilizan versiones anteriores del componente y se utiliza para colocar gráficos de hasta cuatro plumas en los formularios de la aplicación.
- **AxInterop.HMITalkLib.dll / Interop.HMITalkLib.dll / AxInterop.LineChartLib.dll / Interop.LineChartLib.dll**: son archivos requeridos por los objetos HMITalk y LineChart para su funcionamiento cuando se crean aplicaciones en .NET. No son necesarios si se utiliza VB6. Estos archivos se distribuyen con cada driver ya que son utilizados por el utilitario TestHMITalk durante su ejecución. No obstante, se sugiere utilizar los que genera automáticamente el compilador en la carpeta bin\Release de la aplicación. Se debe tener en cuenta que cuando se compila para 32 bits, 64 bits o AnyCPU, estos archivos mantienen los mismos nombres pero sus contenidos son diferentes, por lo que se debe tener la precaución de distribuir las versiones correctas según la CPU de destino.

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

- **RegHMITalk.exe:** utilitario para registrar HMITalk, Grid32 y LineChart. Se encarga de invocar al programa regsvr32.exe de Windows que corresponda a cada plataforma. Si la plataforma lo permite, registra ambas versiones (32 y 64 bits). Se ejecuta automáticamente en modo administrador. Requiere Framework.NET 2.0 instalado para correr. Los objetos OCX pueden ser alternativamente registrados ejecutando manualmente el utilitario RegSvr32.exe de Windows en modo administrador desde una consola de comandos del sistema (DOS).
- **TestHMITalk.exe:** utilitario para probar si los objetos HMITalk y LineChart fueron correctamente registrados. Si se visualizan los controles en los formularios y se los ve funcionando correctamente, esto indica que el registro se realizó correctamente.
- **GetPCId.exe (\*)**: es un utilitario para obtener un número de identificación o PCId de las máquinas cliente. Se puede distribuir a los clientes con el objeto de crear las licencias runtime a partir de los PCId recolectados. Puede ser ejecutado tanto en 32 bits como en 64 bits indistintamente.
- **MakeLic.exe (\*)**: es un utilitario para generar licencias runtime de clientes a partir del número de identificación PCId de la máquina del cliente. Para habilitar su funcionalidad se requiere del ingreso del número de licencia ilimitada que se entrega con el driver adquirido. Se entrega sólo al licenciataro y no debe ser distribuido. Puede ser ejecutado tanto en 32 bits como en 64 bits indistintamente.

(\*) Estos archivos están disponibles únicamente en las versiones licenciadas.

Adicionalmente también se incluyen estos drivers gratuitos con sus manuales:

- **XMODEM:** este driver permite comandar un modem serie o GPRS con secuencia de comandos Hayes AT para realizar un discado previo antes de pasar a comunicarse con el driver XFINS. Permite finalizar la llamada al terminar.
- **XTEST:** este driver permite probar un vínculo, enviando telegramas que espera recibir tal cual han sido transmitidos. El uso de este driver requiere puentear el extremo más alejado del vínculo que se quiere probar. Los mensajes transmitidos son a su vez utilizados como respuesta y contienen valores aleatorios que pueden ser utilizados para animar la aplicación.

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

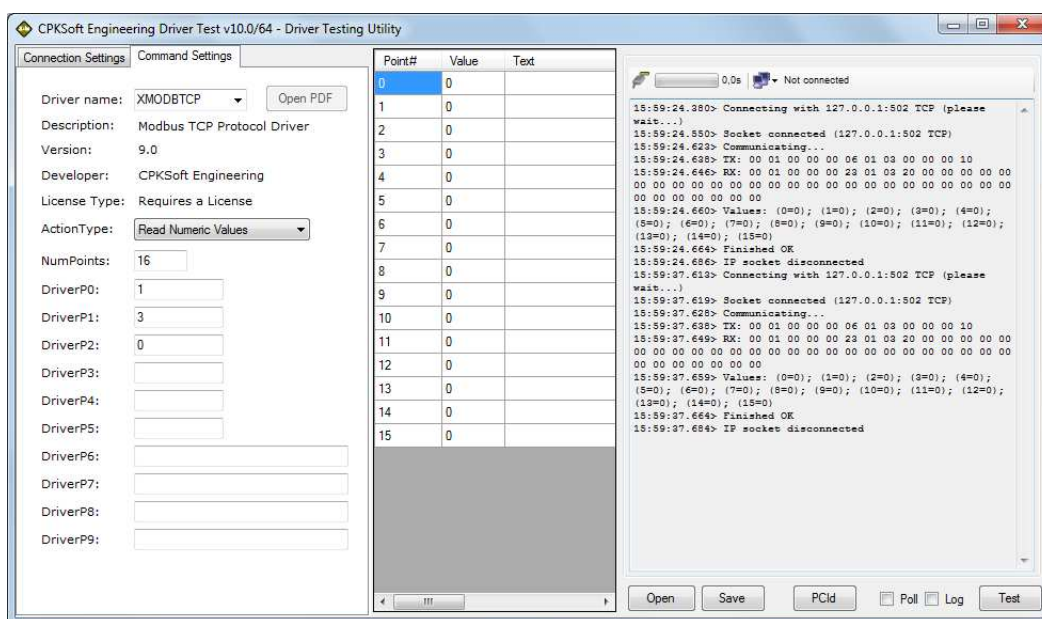
www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

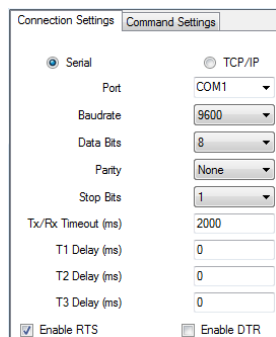
# Soluciones de comunicación industrial para Windows

## Prueba de funcionamiento del driver

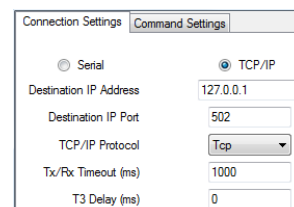
Para ensayar cualquiera de los drivers de CPKSoft Ingeniería se puede utilizar un mismo utilitario llamado **DriverTest.exe** que se distribuye en el archivo .zip de instalación de cada driver. Esta herramienta de testeo permite validar el funcionamiento de todos los comandos ofrecidos por cada driver, así como ensayar diferentes tipos de conexión y seteos de comunicación. No es necesario por lo tanto desarrollar aplicaciones cuando la única finalidad es realizar pruebas de conectividad y viabilidad. La interfaz del programa DriverTest se muestra a continuación (el "/32" ó "/64" luego del número de versión en el título de la ventana del programa indica en qué modo se está ejecutando, si en 32 bits o en 64 bits):



El programa requiere de dos grupos de parámetros que deben ser definidos antes de probar un comando. En primer lugar están los seteos que son propios de la conexión y que establecen en la pestaña **Connection Settings**, como se muestra a continuación:



Configuración de una conexión serie



Configuración de una conexión ethernet

Estos parámetros deben estar en concordancia con la configuración utilizada por el equipo con el cual se esté intentando comunicar. Por ejemplo, en una conexión del tipo serial, la velocidad de comunicación o baudrate que se establezca en el programa DriverTest deberá ser la misma que esté siendo utilizada por el equipo que se pretende acceder.

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

El otro grupo de parámetros que se deben definir son los relacionados con el comando que se quiere probar y se establecen en la pestaña **Command Settings**. En primer lugar se debe definir cuál va a ser el driver que se va a ensayar, seleccionándolo para ello de la lista de drivers disponibles. En esta lista se mostrarán todos aquellos drivers que estén presentes en la carpeta en la que reside el programa DriverTest.exe. Si el driver XFINS es el único instalado en su computadora, será el único driver que le aparecerá en el descolgable.

A continuación, se deben establecer los parámetros **ActionType**, **NumPoints** y **DriverP0** a **DriverP9**, que constituyen la interfaz de configuración común a todos los drivers, y cuyos significados y rangos de valores permitidos se deben consultar para cada comando y driver en particular, en la última parte del manual de cada driver. Por ejemplo, al final de este manual se encuentran todos los comandos ofrecidos por el driver XFINS.

A modo de ejemplo, tomaremos un driver que no necesariamente es el de este manual, pero que a los fines de esta explicación es igualmente ilustrativo. Supongamos que se quisieran leer 16 registros analógicos (numéricos) de un equipo con número de estación 1 que hable el protocolo Modbus TCP, a partir de la dirección 0 de su mapa de memoria. Para realizar esta lectura será necesario entonces utilizar el driver XMODBTCP que es quien sabe manejar el protocolo de comunicaciones Modbus TCP requerido. Si nos remitiéramos a lo especificado en su manual, encontraríamos que el comando que tendríamos que ejecutar es el *Read Holding Registers as Unsigned 16-bit Integers*, ya que su descripción concuerda con la tarea que deseamos realizar.

A continuación se muestra lo indicado en el manual del driver XMODBTCP para este comando, y cómo esa información se utiliza para configurar los parámetros del comando en DriverTest:

**Read Holding Registers as Unsigned 16-bit Integers**

Descripción del comando:  
Obtains the current values in one or more holding registers as unsigned 16-bit integers from 0 to 65535. This command implements Modbus function 3.

Métodos usados para ejecutar este comando:  
ReadNumericValues

Número de puntos permitidos para este comando:  
1-125

Significado del parámetro P0:  
Station number (0-255)

Significado del parámetro P1:  
3

Significado del parámetro P2:  
Indicates the starting register address.

Valores que son devueltos:  
Valor del punto (0) = First register value (0-65535)  
Valor del punto (1) = Second register value (0-65535)  
...  
Valor del punto (n-1) = Last register value (0-65535)

Configuration in DriverTest:

- ActionType: Read Numeric Values
- NumPoints: 16
- DriverP0: 1
- DriverP1: 3
- DriverP2: 0
- DriverP3:
- DriverP4:
- DriverP5:
- DriverP6:
- DriverP7:
- DriverP8:
- DriverP9:

Como se puede ver, para llevar a cabo la función de leer esos 16 registros y de acuerdo a lo que dice el manual del driver XMODBTCP para el comando *Read Holding Registers*, se debe configurar el parámetro ActionType como "ReadNumericValues", el parámetro DriverP0 como 1 (ya que allí se especifica el número de equipo a ser leído), el parámetro DriverP1 como 3 y que es fijo para este comando, el parámetro DriverP2 como 0 (ya que allí se indica la dirección de memoria del primer registro a ser leído y que queremos que sea el 0), y finalmente NumPoints se configura 16 (ya que es la cantidad de registros deseados y está dentro de los límites permitidos por este comando, que según se indica es de 1 a 125). Respecto de los parámetros DriverP3 a DriverP9, se dejan en blanco ya que no son requeridos por este comando.



# Soluciones de comunicación industrial para Windows

Exactamente de igual manera se debe proceder para ensayar cualquier comando de cualquier driver. Es decir, se debe buscar la descripción del comando a ser ensayado en su correspondiente manual, y luego de acuerdo a lo que allí se explique configurar los parámetros del comando en el programa DriverTest.

Una vez configurados los parámetros del comando, se debe presionar el botón Test para iniciar una lectura o escritura.

En caso que se estén leyendo datos, éstos se verán aparecer en la posición que les corresponda en la tabla central. La posición 0 corresponde al primer valor recibido y la posición (NumPoints-1) corresponde al último valor recibido.

En el caso que se estén escribiendo valores, éstos deberán ser previamente ingresados manualmente en la columna "Value" de la planilla, antes de presionar el botón Test.

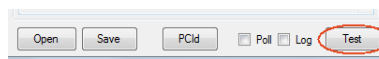
La actividad del driver se puede seguir en el panel de la derecha, donde se registran todos los telegramas transmitidos y recibidos hacia y desde el equipo conectado. El objeto utilizado para construir ese panel no es otro que el mismo NetTalk, que es el que utilizará luego usted como usuario/integrador/desarrollador/programador para crear sus propias soluciones. DriverTest es entonces en realidad una aplicación simple escrita en Visual C#.NET que utiliza NetTalk para manejar los drivers que se quieran probar.

DriverTest puede ser colocado en modo de poleo automático si se activa la casilla Poll. Si se activa la casilla Log, todos los telegramas Tx/Rx de la comunicación se almacenan con timestamp en un archivo ASCII en el disco duro de la máquina, lo que es útil para documentar fallas o para analizar la respuesta de los equipos.

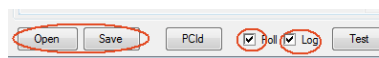
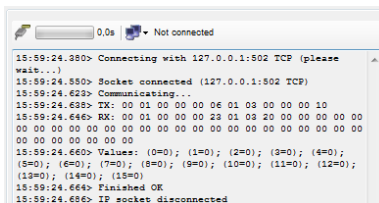
El programa también permite guardar un juego de parámetros de configuración en un archivo .xml para poder ser nuevamente levantado en otro momento y continuar con los ensayos de un comando o utilizarlo como base para un ensayar otro comando bajo condiciones similares.

DriverTest abre y cierra la conexión cada vez que se utiliza el botón Test. Si se deja la opción Poll activada, la conexión se abre con la primera comunicación y se cierra cuando se detiene el poleo al desactivar la casilla.

Los telegramas que se muestran en el panel de comunicaciones se ven completos sólo en el caso de los drivers correctamente licenciados o en el caso de los gratuitos. En los casos de drivers que requieren de licencia pero ésta no se encuentra o no es válida (como puede ser en el caso de una versión de evaluación del driver), los telegramas se mostrarán sólo parcialmente, con signos de interrogación (?) sobre algunos de sus bytes.



Point#	Value	Text
0	0	
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	





# Soluciones de comunicación industrial para Windows

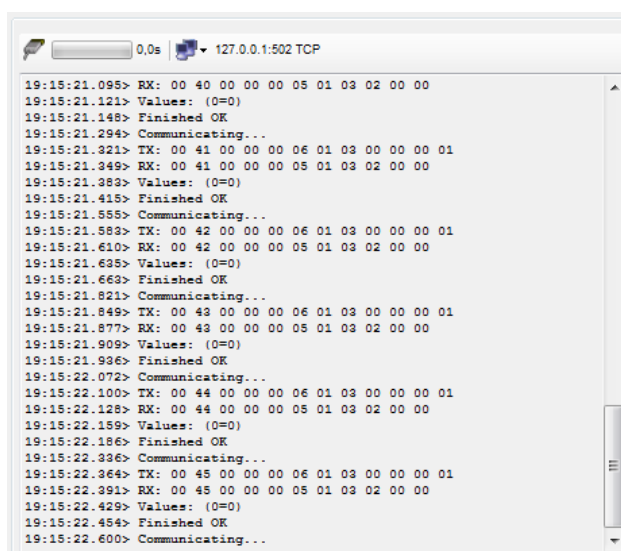
## Uso del componente NetTalk

### La interfaz de NetTalk

El componente NetTalk.dll es un control nativo para .NET cuya finalidad es manejar los drivers de comunicación de CPKSoft Ingeniería desde aplicaciones escritas en .NET. NetTalk permite desarrollar aplicaciones que utilicen Framework.NET 2.0 en adelante.

Cada instancia de NetTalk posee su propia interfaz gráfica que se puede colocar en tiempo de diseño en los formularios (WinForm) de las aplicaciones creadas con VisualStudio.NET. Las instancias de NetTalk se pueden también crear dinámicamente de manera oculta agregando una referencia a la librería o ensamblado en la aplicación y creando dichas instancias en tiempo de ejecución.

NetTalk ofrece una amplia lista de propiedades, métodos y eventos para establecer todos los aspectos funcionales necesarios para un manejo completo de la apariencia del objeto y de los drivers de comunicación.



NetTalk ofrece una interfaz que permite visualizar en tiempo real todas las comunicaciones que tienen lugar a través de la conexión que está activa.

Cada telegrama tiene una marca de tiempo o timestamp asociada, con resolución al milisegundo.

Los telegramas enviados hacia los equipos están precedidos por la leyenda TX.

Los telegramas recibidos desde los equipos están precedidos por la leyenda RX.

El formato por defecto para visualizar los telegramas es el hexadecimal, donde cada byte transmitido o recibido se muestra como un valor entre 00 y FF. Este formato se puede modificar con la propiedad PanelMode a ASCII o a Decimal.

Varias propiedades y métodos alteran la apariencia del objeto, como la propiedad PanelLines que establece cuántas líneas de historia de comunicación almacena la ventana.

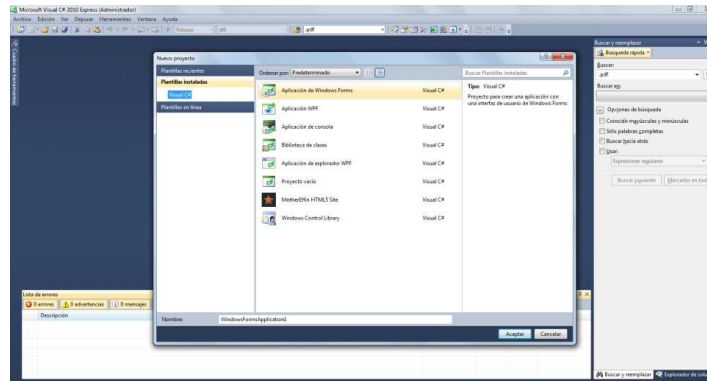
Más adelante en este manual se da la lista completa de propiedades, métodos y eventos.

### Crear una aplicación

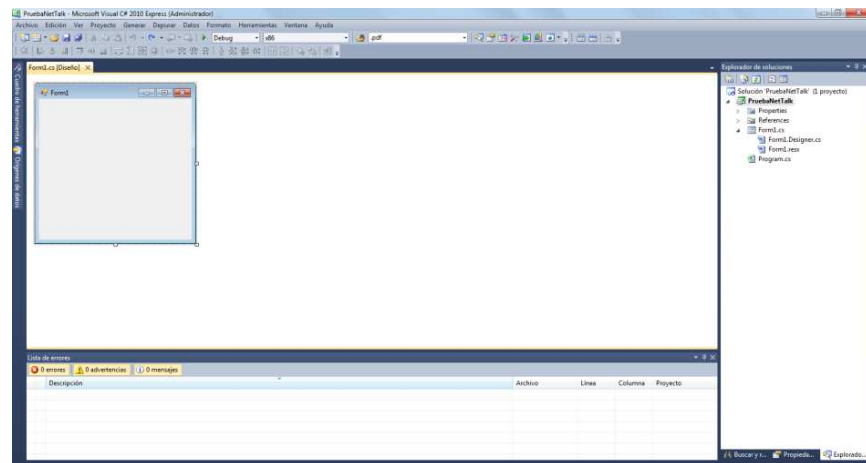
Las aplicaciones .NET se pueden escribir en varios lenguajes .NET, siendo los más comunes Visual Basic y C#. Para mostrar cómo crear una aplicación utilizaremos Microsoft Visual C# 2010 Express, que es una herramienta totalmente gratuita y se puede descargar desde la página de Microsoft. Visual Basic y C# son lenguajes de aspecto bastante similar entre sí y todo lo explicado para los ejemplos de este manual que están escritos en C#, puede ser fácilmente trasladado a Visual Basic si prefiere trabajar con ese lenguaje.

Para crear una aplicación, el primer paso es ir a la opción de menú *Archivo+Nuevo proyecto...* y allí seleccionar *Aplicación de Windows Forms*, como muestra la imagen:

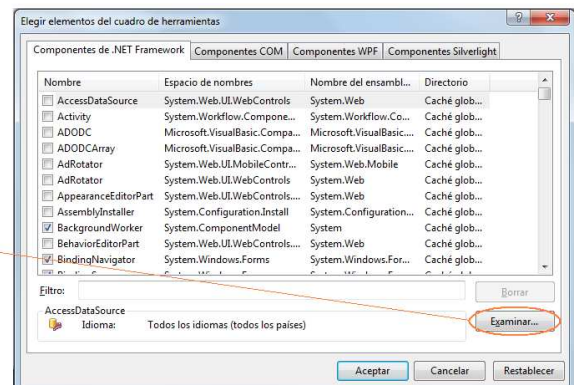
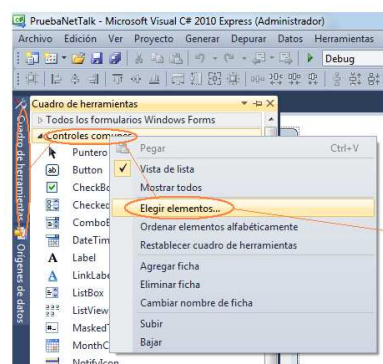
# Soluciones de comunicación industrial para Windows



Una vez establecido el nombre de la aplicación aceptamos, tras lo cual aparecerá un formulario vacío. En el caso de nuestro ejemplo, hemos llamado al proyecto “PruebaNetTalk”:



Para poder agregar el objeto NetTalk, debemos antes agregarlo al cuadro de herramientas de Visual Studio. Para ello deberemos posicionarnos sobre el *Cuadro de herramientas* y sobre *Controles comunes* presionamos botón derecho del mouse y aparecerá un menú descolgable. Allí seleccionamos la opción *Elegir elementos...* y nos aparecerá una ventana con un botón *Examinar...* que nos permitirá buscar el archivo NetTalk.dll en nuestro disco rígido:



CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

## pág. 11/58

# Soluciones de comunicación industrial para Windows

Otras propiedades menos usadas pero que también suelen manejar desde el arranque son:

- `this.netTalk1.OnErrorReconnect`
- `this.netTalk1.PanelLines`
- `this.netTalk1.PanelFont`
- `this.netTalk1.SilentMode`
- `this.netTalk1.MinimizeDoEventsCalls`
- `this.netTalk1.FreezeWhenDoubleClick`
- `this.netTalk1.EnableShowHidePanelButton`
- `this.netTalk1.EnableAbortCommunicationButton = false;`
- `this.netTalk1.EnableCloseConnectionButton = false;`
- `this.netTalk1.EnableReconnectButton`

Puede encontrar más adelante al significado de cada propiedad una en la sección sobre “Propiedades”.

## Manejo de la conexión

Antes de poder utilizar un driver para comunicarse con cualquier equipo, es necesario abrir una conexión, ya sea ésta del tipo serial o del tipo tcp/ip (ethernet), lo que dependerá del canal a través del cual se va a comunicar. Esta acción, como la mayoría de las acciones que se pueden realizar con el objeto, se debe implementar mediante la escritura de código.

La conexión necesita abrirse una única vez y mientras la misma permanezca abierta, se puede realizar cualquier número de llamadas de lectura o escritura al objeto, incluso cambiando de driver entre llamada y llamada. Los parámetros con los que se abrió la conexión permanecerán constantes hasta que ésta se cierre y eventualmente se vuelva a abrir con otros parámetros.

Si se desea dejar libre el recurso utilizado por la conexión (por ejemplo liberar un determinado puerto serial para poder utilizarlo desde otros programas) entre llamadas sucesivas de lectura o escritura, se puede abrir y cerrar la misma cada vez que se realice una comunicación, aunque esto agregará un tiempo extra a cada operación de lectura o escritura. También la conexión se puede cerrar y volver a abrir si se desea cambiar alguno de sus parámetros entre una llamada y otra (por ejemplo, si se quiere cambiar la velocidad de una comunicación serial).

## Conexión serie

Para abrir una conexión serie se utiliza el método **Connect**, pasándole los siguientes parámetros relacionados con el puerto serial:

- **PortName:** Nombre del puerto serie (“COM1”, “COM2”, etc.).
- **BaudRate:** Velocidad de comunicación o baudiaje (4800, 9600, 19200, etc.).
- **Parity:** Paridad (None para “ninguna”, Even para “par”, Odd para “impar”).
- **DataBits:** Bits de datos (7 u 8).
- **StopBits:** Bits de stop (1, 1.5 o 2).
- **Timeout(\*):** Tiempo máximo de espera de respuesta (en milisegundos, típico 2000).
- **T1Delay:** Tiempo desde encendido de RTS hasta inicio de transmisión (en milisegundos).
- **T2Delay:** Tiempo desde fin de transmisión hasta apagado de RTS (en milisegundos).
- **T3Delay:** Tiempo de silencio forzado desde fin de recepción (en milisegundos).
- **RtsEnable:** Indicador de si se debe activar RTS durante la comunicación (\*\*).
- **DtrEnable:** Indicador de si se debe activar DTR durante la comunicación.

(\*) Se debe tener la precaución de establecer un Timeout lo suficientemente grande como para darle al equipo contactado tiempo a preparar y despachar su respuesta. Caso contrario, el driver puede arrojar errores de timeout (sin respuesta/no answer) cuando un equipo responde el pedido correctamente, pero fuera de ese tiempo.

(\*\*) Se recomienda dejar el RTS activado siempre que se utilice un conversor de RS-232 a RS-422/485.

La declaración formal del método es la siguiente:

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

```
Boolean Connect(  
    String PortName,  
    Int32 BaudRate,  
    Parity Parity,  
    Int32 DataBits,  
    StopBits StopBits,  
    Int32 TimeOut,  
    Int32 T1Delay,  
    Int32 T2Delay,  
    Int32 T3Delay,  
    Boolean RtsEnable,  
    Boolean DtrEnable)
```

En el código siguiente se muestra un ejemplo de llamado a esta función:

```
this.netTalk1.Connect("COM2", 9600, System.IO.Ports.Parity.None, 8,  
System.IO.Ports.StopBits.One, 2000, 0, 0, 0, true, false);
```

Esta llamada abre una conexión serial a través del puerto serie COM2, a 9600 baudios, sin paridad, con 8 bits de datos y 1 bit de stop, con 2000 milisegundos de timeout de espera, sin definir tiempos de espera de RTS ni tiempo de silencio, activando RTS y sin activar DTR.

## Conexión tcp/ip

Para abrir una conexión tcp/ip o ethernet, se utiliza una sobrecarga del mismo método **Connect**, pero esta vez pasándole parámetros relacionados con la conexión IP:

- **Address:** Dirección IP de destino
- **PortNum:** Puerto IP de destino
- **Protocol:** Protocolo IP a utilizar (TCP, UDP)
- **Timeout(\*):** Tiempo máximo de espera de respuesta (en milisegundos, típico 2000).
- **T3Delay:** Tiempo de silencio forzado desde fin de recepción (en milisegundos).

(\*) Para este parámetro son válidas las mismas consideraciones que para el caso de una conexión serial.

La declaración formal del método es la siguiente (sobrecarga del caso serial):

```
Boolean Connect(  
    String Address,  
    Int32 PortNum,  
    TcpIpProtocolType Protocol,  
    Int32 TimeOut,  
    Int32 T3Delay)
```

En el código siguiente se muestra un ejemplo de llamado a esta función:

```
this.netTalk1.Connect("95.154.201.59", 8010, NetTalk.TcpIpProtocolType.Tcp, 2000,  
0);
```

Esta llamada abre una conexión tcp/ip mediante un socket IP al puerto 8010 de la dirección IP 95.154.201.59, utilizando protocolo TCP, con 2000 milisegundos de timeout de espera y sin tiempo de silencio al final.

## Código de conexión típico

Un código típico para abrir una conexión desde un botón es el siguiente:

```
private void btnAbrirConexion_Click(object sender, EventArgs e)  
{  
    if (!this.netTalk1.IsConnected)
```

# Soluciones de comunicación industrial para Windows

```
{
    if (!this.netTalk1.Connect("127.0.0.1", 502, NetTalk.TcpIpProtocolType.Tcp,
2000, 0))
    {
        MessageBox.Show("Error al abrir conexión: " + this.netTalk1.Status);
    }
}
else MessageBox.Show("Ya hay una conexión abierta.");
}
```

En este código se agrega un chequeo previo de conexión “ya abierta” antes de intentar abrirla, consultando para ello la propiedad **IsConnected** antes de llamar al método **Connect** (el símbolo “!” utilizado en el “if” funciona igual que la palabra “Not” de Visual Basic). Si la conexión ya se encontraba abierta, se muestra un mensaje de aviso y el código no hace mas nada. Si no había ninguna conexión abierta, se llama al método **Connect**, quien devuelve **true** si la conexión se pudo abrir OK y **false** si hubo algún error. En caso de error, se muestran los detalles del mismo mediante la propiedad **Status**.

## Cierre de la conexión

No es necesario cerrar una conexión luego de cada comunicación, en la medida que dicha conexión siga siendo válida para el próximo equipo a ser comunicado. Por ejemplo, si se va a seguir interrogando un mismo equipo o bien si se va a interrogar otro equipo pero que comparte el mismo puerto serial que el anterior, entonces se puede seguir utilizando la misma conexión ya abierta.

Si ese va a comunicar con otro equipo que tiene un puerto serial o una dirección tcp/ip diferente, entonces será necesario cerrar la conexión anterior y reabrirla con los nuevos parámetros necesarios.

La forma de cerrar la conexión es mediante una llamada al método **Disconnect**, como se muestra en el ejemplo:

```
private void btnCerrarConexion_Click(object sender, EventArgs e)
{
    if (this.netTalk1.IsConnected)
    {
        if (!this.netTalk1.Disconnect())
        {
            MessageBox.Show("Error al cerrar conexión: " + this.netTalk1.Status);
        }
    }
    else MessageBox.Show("No hay una conexión abierta.");
}
```

## Manejo de lecturas

Una vez abierta una conexión, ésta se utilizará para canalizar todas las lecturas que se realicen de ahí en más, hasta que se la cierre.

Las lecturas se realizan mediante llamados a los métodos **ReadNumericValues** y **ReadBooleanValues** (según se trate de valores analógicos o discretos), en los que se le pasa al **NetTalk** el nombre del driver y los parámetros relativos al comando específico que se quiere ejecutar. Estos parámetros son la cantidad de puntos o valores a ser leídos, los parámetros **DriverP0** a **DriverP9** propios del comando y un flag que indica si la llamada al método debe esperar o no a terminar la comunicación con el equipo antes de devolver el control a la aplicación.

El comando **ReadNumericValues** se utiliza cuando los valores que se espera recibir desde el equipo son del tipo numérico o analógico, es decir, valores del tipo 0, 1, 2, 3, etc., o bien valores en punto flotante, tanto positivos como negativos. Se utiliza típicamente para consultar valores de registros enteros, timers, contadores, entradas analógicas, valores en punto flotante (como puede ser el caso de mediciones eléctricas), etc.



# Soluciones de comunicación industrial para Windows

El comando ReadBooleanValues se utiliza cuando los valores que se espera recibir desde el equipo son del tipo bit o booleano, es decir, estados ON/OFF, 0/1, etc., que son típicos cuando se leen entradas digitales, estados de bobinas, etc.

Las declaraciones formales de estos métodos son iguales:

```
public Boolean ReadNumericValues/ReadBooleanValues(  
    String DriverName,  
    Int32 NumPoints,  
    String P0,  
    String P1,  
    String P2,  
    String P3,  
    String P4,  
    String P5,  
    String P6,  
    String P7,  
    String P8,  
    String P9,  
    bool Wait)
```

El parámetro **DriverName** indica el nombre del driver que el método debe utilizar para realizar la comunicación. Para el driver al que se refiere este manual, el nombre que se utilice en este parámetro puede ser directamente el texto "XFINS", en cuyo caso NetTalk buscará por defecto el archivo XFINS.xx.tlk en la carpeta en la que esté alojado el ensamblado NetTalk.dll, donde "xx" será el modo de ejecución (32 bits o 64 bits) de la aplicación en ese momento. Se puede también agregar el camino completo al archivo .tlk del driver a ser utilizado, como por ejemplo "c:\MiAplicación\Drivers\XFINS.32.tlk" (si la plataforma de destino va a ser de 32 bits)", "c:\MiAplicación\Drivers\XFINS.64.tlk" (si la plataforma de destino va a ser de 64 bits)" o también "c:\MiAplicación\Drivers\XFINS". En este último caso, cuando se indica el nombre del driver sin su extensión, NetTalk lo completa automáticamente con la extensión correcta.

El parámetro **NumPoints** indica la cantidad de puntos o valores que se desea leer y siempre debe estar dentro de los límites admitidos por el comando específico que se vaya a ejecutar.

Los parámetros **P0** a **P9** suministran al driver toda la información requerida por el comando que se debe ejecutar. Esa información se obtiene de la última parte del manual de cada driver, como ya se adelantó en la sección dedicada al programa DriverTest.

El parámetro **Wait** define si la lectura se realizará con bloqueo o sin bloqueo de la ejecución del código. Estos dos tipos de lectura se explican en detalle más adelante.

Tanto en el caso de entradas numéricas como booleanas, una vez realizada la comunicación los valores recibidos quedan disponibles para la aplicación a través del método **PointValue(PointIndex)**, donde *PointIndex* es el subíndice del valor obtenido. Este subíndice toma el valor 0 para el primer valor y (NumPoints-1) para el último valor leído, donde NumPoints es la cantidad de puntos o valores que se le solicitaron al comando del driver.

Ambos métodos devuelven un valor del tipo booleano que es **true** cuando la llamada al método fue exitosa y es **false** cuando hubo algún error. Como se verá más adelante, las situaciones en las que estos métodos devuelven true o false dependen en realidad de si hay o no hay bloqueo en la ejecución.

## Lecturas con espera

Cuando el parámetro **Wait** se establece a **true** las lecturas se realizan con "espera" o "bloqueo de ejecución", también llamadas "lecturas sincrónicas", lo que significa que al llamar a cualquiera de los métodos ReadNumericValues o ReadBooleanValues, la ejecución del código se detendrá a la espera de que se termine toda la comunicación necesaria con el equipo. La tarea que resultará bloqueada es aquella desde la que se invocó al método de lectura. La aplicación puede mientras tanto mantener otras tareas en ejecución sin problemas.

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354



# Soluciones de comunicación industrial para Windows

Al ejecutarse el método de lectura, se inicia un intercambio de telegramas entre el driver y el equipo. Mientras la ejecución del código permanece detenida, el flujo de esta comunicación es manejado en background por el propio objeto NetTalk, quien va realizando internamente llamadas al driver pidiéndole que éste envíe telegramas hacia el equipo o que analice los telegramas recibidos desde el equipo, hasta que se haya cumplido con todas las transacciones requeridas por el comando solicitado o bien se haya producido un error de timeout.

Este tipo de lectura con bloqueo permite escribir código del tipo lineal o sincrónico, es decir, que se vaya ejecutando línea por línea siguiendo una secuencia clara y predecible. En ciertas aplicaciones, donde las lecturas de los equipos se deben ir haciendo de manera controlada, sincronizada, según una cierta secuencia o según una cierta dependencia de tipo cascada, lecturas a demanda, etc., este modo de lectura es el más indicado.

En las lecturas con bloqueo, los métodos ReadNumericValues y ReadBooleanValues devuelven **true** cuando ejecutaron correctamente la lectura solicitada y por lo tanto, cuando los valores leídos ya se encuentran a disposición de la aplicación para ser levantados a través del método PointValue().

Cuando haya incongruencias o valores inválidos en los parámetros utilizados durante las llamadas a los métodos, o bien cuando se hayan producido errores de comunicación de algún tipo (error de timeout porque el equipo no respondió a tiempo, conexión sin abrir, conexión caída, etc.), los métodos devolverán **false** y el detalle del error estará disponible en la propiedad **Status** del componente NetTalk. En este caso, se deberán ignorar los valores que devuelva el método PointValue(). Al final del manual se da una lista de los mensajes de error posibles durante una comunicación cuando se utiliza el driver XFINS.

El tiempo que permanecerá la ejecución detenida mientras dure la llamada al método dependerá del tiempo que tome la ejecución del comando del driver en ser completada. Puede variar desde algunas centésimas de segundo hasta varios segundos o incluso minutos, según por ejemplo se esté pidiendo un simple registro por tcp/ip o bien se estén descargando todos los eventos almacenados en la memoria de un equipo por un puerto serial. En cualquier caso, si se produjera una falla en la comunicación y el equipo nunca respondiera o habiéndolo hecho, dejara de responder, el bloqueo terminará automáticamente pasado el tiempo especificado en el parámetro **Timeout** de la conexión.

Un ejemplo de una lectura con bloqueo desde un botón es la siguiente:

```
private void btnLeerConBloqueo_Click(object sender, EventArgs e)
{
    if (this.netTalk1.ReadNumericValues("XMODBTCP", 16, "1", "3", "0", "", "",
        "", "", "", "", "", true))
    {
        MessageBox.Show("Valor del registro 0: " +
            this.netTalk1.PointValue(0).ToString());
    }
    else MessageBox.Show("Error leyendo registro: " + this.netTalk1.Status);
}
```

En este ejemplo se utiliza el driver XMODBTCP para preguntar a un equipo que habla Modbus TCP y con número de estación 1 por el estado de 16 registros enteros a partir de la dirección 0 de su mapa de memoria. Es el mismo caso que ya se utilizó como ejemplo cuando se explicó el utilitario DriverTest al principio del manual, con la diferencia que los mismos datos que antes se cargaban en la pestaña de *Command Settings* ahora aquí se le pasan como argumento en la llamada al método ReadNumericValues.

La ejecución con bloqueo queda clara en este ejemplo donde el método ReadNumericValues se utiliza en una sentencia "if" y donde inmediatamente terminada la ejecución del método, se puede mostrar con tranquilidad el valor de los registros recibidos. Sólo se está mostrando un mensaje con el valor del primer registro recibido, utilizando para ello la llamada al método PointValue pasándole el parámetro 0. Se observa también que, ante un error (en la parte "else" del "if"), la rutina muestra un mensaje con la propiedad Status para describir el error que se produjo.

# Soluciones de comunicación industrial para Windows

## Lecturas por eventos

Cuando el parámetro **Wait** se establece a **false**, las lecturas se realizan sin esperar a que la comunicación con el equipo se termine de realizar y por lo tanto el método invocado devuelve el control a la aplicación de forma inmediata y el código nunca se bloquea. Se conocen también como lecturas asincrónicas y la característica es que no hay bloqueo en la ejecución mientras se espera que lleguen los datos de respuesta. La finalidad del método en este caso es la de encolar el pedido en el objeto NetTalk, quien se hará cargo internamente de procesar toda la comunicación, avisando luego cuando la ésta haya terminado mediante el disparo del evento Finished. La ejecución en este caso no es del tipo lineal sino que el código se ejecuta de manera asincrónica o por eventos, sin un hilo de ejecución totalmente predecible.

El tipo de lectura sin bloqueo puede ser apropiado cuando se va a tener un poleo o barrido permanente y no se desea que la aplicación se “frene” en cada lectura, dando la sensación de que la aplicación responde “de a ratos” a las órdenes del usuario. No obstante, un código donde se hagan lecturas con bloqueo pero desde una tarea corriendo en background, también puede servir perfectamente para realizar un barrido continuo de equipos sin afectar la respuesta de la aplicación.

En las lecturas sin bloqueo, los métodos ReadNumericValues y ReadBooleanValues devuelven **true** cuando encolaron correctamente la lectura solicitada, lo que no quiere decir que la comunicación ya se haya realizado y que los datos leídos ya estén disponibles para ser accedidos a través del método PointValue(). Cuando ReadNumericValues y ReadBooleanValues devuelven **false**, están indicando un error al encolarse el pedido, cuya razón debe consultarse en la propiedad Status.

Para poder levantar con seguridad los datos leídos, habrá que esperar a que el método **Finished** se haya disparado y verificar que el argumento **finishedok** haya sido **true**. Si el argumento recibido fuera **false**, quiere decir que se produjo algún tipo de error durante la lectura y por lo tanto habrá que ignorar los datos devueltos por el método PointValue y consultar la propiedad Status para averiguar el detalle del error.

A continuación se muestra la misma lectura de 16 registros del ejemplo utilizado para el caso con bloqueo, adaptado a un caso sin bloqueo:

```
private void btnLeerSinBloqueo_Click(object sender, EventArgs e)
{
    if (!this.netTalk1.ReadNumericValues("XMODBTCP", 16, "1", "3", "0", "", "",
    "", "", "", "", "", false))
    {
        MessageBox.Show("Error encolando pedido: " + this.netTalk1.Status);
    }
}

private void netTalk1_Finished(object sender, bool finishedok)
{
    if (finishedok)
    {
        MessageBox.Show("Valor del registro: " +
        this.netTalk1.PointValue(0).ToString());
    }
    else MessageBox.Show("Error leyendo registro: " + this.netTalk1.Status);
}
```

Aquí se ve como la llamada al método ReadNumericValues simplemente devuelve el control a la aplicación sin realizar ninguna otra acción, salvo la de mostrar un mensaje con la propiedad Status en caso que no se haya podido encolar correctamente el pedido.

La acción de mostrar el valor recibido queda ahora delegada al evento Finished, quien se disparará una vez haya finalizado la comunicación, ya sea que ésta lo haya hecho de manera exitosa o que haya habido un error durante la misma. Mientras no se dispare el evento Finished, la tarea que llamó al método ReadNumericValues quedará libre de ejecutar otras líneas de código o de atender otros eventos que se disparen en la aplicación.

CPKSoft Ingeniería

Drivers para comunicación industrial.

www.cpksoft.com.ar

www.facebook.com/

cpksoftingenieria

cpksoftingenieria@

hotmail.com

tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

El ejemplo también muestra como al dispararse el evento **Finished**, se chequea el argumento **finishedok** para determinar si se puede mostrar el primer valor recibido o bien si se debe mostrar un mensaje de comunicación fallida.

## Manejo de escrituras

El manejo de escrituras con NetTalk es muy similar al manejo de las lecturas. En este caso también existen las escrituras con y sin bloqueo y ambos tipos se manejan de manera idéntica a todo lo explicado anteriormente para las lecturas. Los métodos que se utilizan en este caso son `WriteNumericValues` y `WriteBooleanValues`, dependiendo del tipo de dato a ser enviados al equipo.

La única diferencia entre lecturas y escrituras, es que en el caso de escrituras, los valores a ser enviados se deben cargar mediante el método `PointValue` previamente a la llamada al método de escritura que se utilice.

Sintetizando esta diferencia:

- Lecturas: SE LLAMA AL METODO DE LECTURA -> SE LEEN LOS POINTVALUE()
- Escrituras: SE ESCRIBEN LOS POINTVALUE() -> SE LLAMA AL METODO DE ESCRITURA

El siguiente ejemplo muestra cómo utilizar el driver `XMODBTCP` para escribir un valor 100 en un único registro en la dirección 0 de un equipo que habla Modbus TCP y con número de estación 1:

```
private void btnEscritura_Click(object sender, EventArgs e)
{
    this.netTalk1.PointValue(0, 100);
    if (this.netTalk1.WriteNumericValues("XMODBTCP", 1, "1", "6", "0", "", "",
    "", "", "", "", "", true))
    {
        MessageBox.Show("Registro escrito OK");
    }
    else MessageBox.Show("Error escribiendo registro: " + this.netTalk1.Status);
}
```

En este caso se utiliza una escritura con bloqueo y se puede ver cómo se carga previamente el valor 100 en el `PointIndex 0` utilizando el método `PointValue`. Tras la escritura, se muestra un mensaje indicando que la misma se realizó correctamente, o bien con error. Es relativamente sencillo adaptar este ejemplo a un caso de escritura sin bloqueo como el que ya fue explicado para las lecturas sin bloqueo.

Las explicaciones dadas hasta aquí son genéricas y aplicables a cualquier driver y hemos utilizado al driver `Modbus TCP` como base en todos los ejemplos por ser uno de los más populares en la industria. Para el caso en que se utilice cualquier otro driver que no sea el `Modbus TCP`, deben seguirse los mismos lineamientos ya explicados en este manual. La única diferencia que va a existir en la implementación de comandos de otros drivers es que la configuración de los parámetros `NumPoints` y `DriverP0` a `DriverP9` deberán respetar los rangos y valores que se expliquen para cada comando en el manual del driver que se utilice.

## Interfaz COM del componente NetTalk

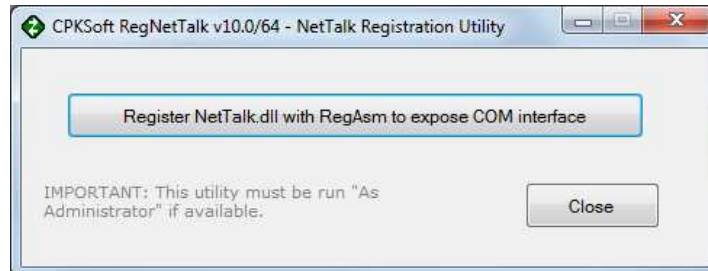
NetTalk posee una interfaz de tipo COM que puede ser expuesta para plataformas de desarrollo que soporten esta tecnología, como es el caso de Visual Basic 6.0. En este caso, es necesario registrar previamente el ensamblado mediante el utilitario `RegNetTalk.exe` que se provee en el `.zip` de instalación:

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows



También se puede registrar el objeto utilizando la herramienta de registro de ensamblados RegAsm.exe que está presente en Windows. El utilitario se puede encontrar en esta carpeta:

C:\Windows\Microsoft.NET\Framework\v2.0.50727

Para sistemas de 64 bits, se utiliza el que está en esta carpeta:

C:\Windows\Microsoft.NET\Framework64\v2.0.50727

La línea de comando, que se debe ejecutar desde una consola de comandos del sistema disparada en modo administrador, es la siguiente:

```
Regasm.exe nettalk.dll
```

El siguiente código muestra cómo crear dinámicamente un objeto NetTalk en un formulario de VB6:

```
Private Sub Command1_Click()  
    Set nt = Controls.Add("NetTalk.NetTalk", "nt", Me)  
    nt.Left = 100  
    nt.Width = Me.Width - 400  
    nt.Top = 100  
    nt.Height = Me.Height - 700  
    nt.Visible = True  
    nt.object.Language = NetTalk.Languages_Castellano  
End Sub
```

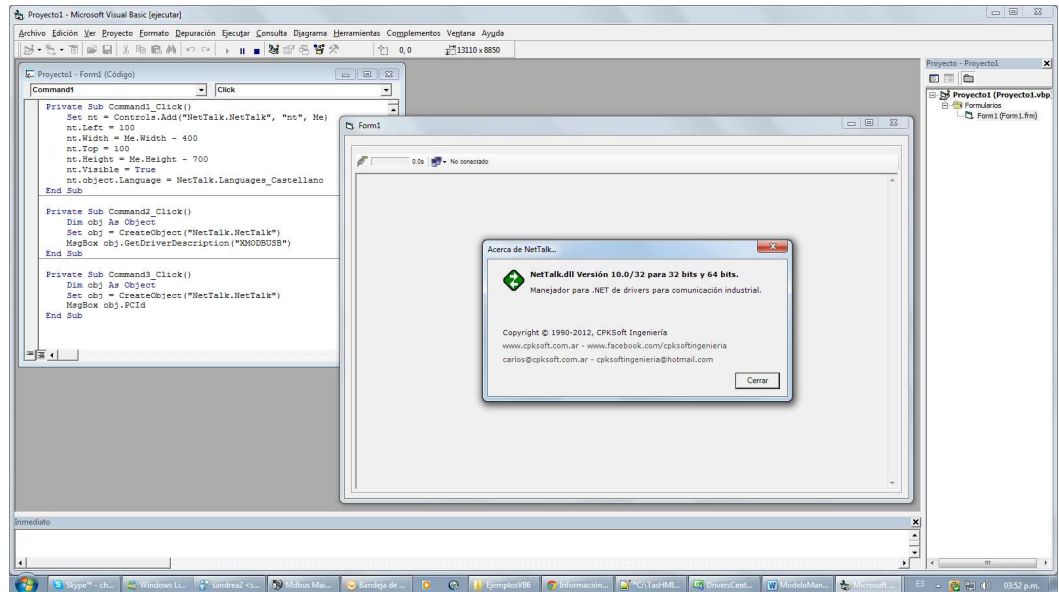
En la siguiente imagen se muestra una imagen del formulario de VB6 con el objeto NetTalk creado en runtime:

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows



Este otro código crea un objeto NetTalk y llama al método que devuelve la descripción de un driver:

```
Private Sub Command2_Click()  
    Dim obj As Object  
    Set obj = CreateObject("NetTalk.NetTalk")  
    MsgBox obj.GetDriverDescription("XMODBUS")  
End Sub
```

Al ejecutarlo, produce la siguiente salida:



Este código llama a la propiedad PCId para conocer el número de máquina en la que se ejecuta el objeto:

```
Private Sub Command3_Click()  
    Dim obj As Object  
    Set obj = CreateObject("NetTalk.NetTalk")  
    MsgBox obj.PCId  
End Sub
```

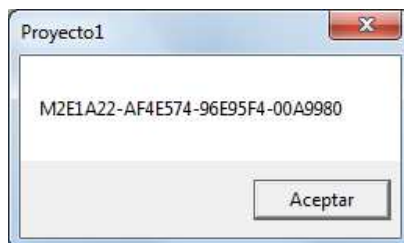
El resultado es un mensaje con el número PCId de la máquina:

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows



## Guía rápida de propiedades, métodos y eventos

### Propiedades

Nombre de la propiedad	Tipo	Descripción
<b>CheckLicense</b>	Short	Si es true, indica que el driver requiere de una licencia. Si es false, indica que el driver es gratuito.
<b>CommBaudRate</b>	Int32	Velocidad de comunicaciones utilizada en la última conexión serie abierta.
<b>CommDataBits</b>	Int32	Bits de datos utilizados en la última conexión serie abierta.
<b>CommDtrEnable</b>	Boolean	DtrEnable utilizado en la última conexión serie abierta.
<b>CommParity</b>	Parity	Paridad utilizada en la última conexión serie abierta.
<b>CommPortName</b>	String	Puerto de comunicaciones utilizado en la última conexión serie abierta.
<b>CommRtsEnable</b>	Boolean	RtsEnable utilizado en la última conexión serie abierta.
<b>CommStopBits</b>	StopBits	Bits de stop utilizados en la última conexión serie abierta.
<b>CommT1Delay</b>	Int32	Retardo T1 utilizado en la última conexión serie abierta.
<b>CommT2Delay</b>	Int32	Retardo T2 utilizado en la última conexión serie abierta.
<b>CommT3Delay</b>	Int32	Retardo T3 utilizado en la última conexión serie abierta.
<b>CommTimeOut</b>	Int32	Timeout utilizado en la última conexión serie abierta.
<b>DateTimeFormat</b>	String	Formato de fecha y hora en el panel de comunicaciones. Por defecto se utiliza "HH:mm:ss.fff". Si se deja vacío, no se muestra el timestamp en el panel.
<b>DriverDescription</b>	String	Descripción del driver utilizado en el último llamado de lectura o escritura.
<b>DriverDeveloper</b>	String	Desarrollador del driver utilizado en el último llamado de lectura o escritura.
<b>DriverLocked</b>	Boolean	Si es true, indica que aún no se validó la licencia del driver utilizado en el último llamado de lectura o escritura. Si es false, la licencia ya fue validada OK.
<b>DriverName</b>	String	Nombre del driver utilizado en el último llamado de lectura o escritura.
<b>DriverNumPoints</b>	Int32	Cantidad de puntos utilizados en el último llamado de lectura o escritura.
<b>DriverP0</b>	String	Parámetro P0 utilizado en el último

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

<b>DriverP1</b>	String	llamado de lectura o escritura. Parámetro P1 utilizado en el último llamado de lectura o escritura.
<b>DriverP2</b>	String	Parámetro P2 utilizado en el último llamado de lectura o escritura.
<b>DriverP3</b>	String	Parámetro P3 utilizado en el último llamado de lectura o escritura.
<b>DriverP4</b>	String	Parámetro P4 utilizado en el último llamado de lectura o escritura.
<b>DriverP5</b>	String	Parámetro P5 utilizado en el último llamado de lectura o escritura.
<b>DriverP6</b>	String	Parámetro P6 utilizado en el último llamado de lectura o escritura.
<b>DriverP7</b>	String	Parámetro P7 utilizado en el último llamado de lectura o escritura.
<b>DriverP8</b>	String	Parámetro P8 utilizado en el último llamado de lectura o escritura.
<b>DriverP9</b>	String	Parámetro P9 utilizado en el último llamado de lectura o escritura.
<b>DriverStatus</b>	String	Idem propiedad Status.
<b>DriverVersion</b>	String	Versión del driver utilizado en el último llamado de lectura o escritura.
<b>EnableAbortCommunicationButton</b>	Boolean	Establece si estará disponible la opción 'Interrumpir Comunicación' en el menú descolgable.
<b>EnableClearPanelButton</b>	Boolean	Establece si estará disponible la opción 'Limpiar Panel' en el menú descolgable.
<b>EnableCloseConnectionButton</b>	Boolean	Establece si estará disponible la opción 'Cerrar Conexión' en el menú descolgable.
<b>EnableFreezeUnfreezePanelButton</b>	Boolean	Establece si estará disponible la opción 'Congelar/Descongelar Panel' en el menú descolgable.
<b>EnableReconnectButton</b>	Boolean	Establece si estará disponible la opción 'Reconectar' en el menú descolgable.
<b>EnableSavePanelButton</b>	Boolean	Establece si estará disponible la opción 'Guardar Panel' en el menú descolgable.
<b>EnableSetLinesToSaveButton</b>	Boolean	Establece si estará disponible la opción 'Establecer cantidad de líneas' en el menú descolgable.
<b>EnableShowHidePanelButton</b>	Boolean	Establece si estará disponible la opción 'Ocultar/Mostrar Panel' en el menú descolgable.
<b>FreezePanel</b>	Boolean	Si es true el panel se congela y si es false el panel se descongela.
<b>FreezeWhenDoubleClick</b>	Boolean	Establece si el panel se puede congelar/descongelar haciendo doble-click con el mouse sobre él.
<b>IsCommunicating</b>	Boolean	Indica si el driver se está comunicando en ese momento.
<b>IsConnected</b>	Boolean	Indica si hay una conexión abierta en ese momento.
<b>Language</b>	Languages	Establece el idioma que se mostrará en la interfaz y en los mensajes propios del componente.
<b>LastDriverStatus</b>	String	Muestra un mensaje con el último estado retornado por el driver.
<b>LastException</b>	String	Muestra un mensaje con la última excepción que se produjo en el

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/cpksoftingenieria](http://www.facebook.com/cpksoftingenieria)

[cpksoftingenieria@hotmail.com](mailto:cpksoftingenieria@hotmail.com)

tel: 54-11-1545788354



# Soluciones de comunicación industrial para Windows

<b>LicenseType</b>	String	componente. Tipo de licencia del driver utilizado en el último llamado de lectura o escritura.
<b>LogPanel</b>	Boolean	Establece si se registra en disco la actividad del panel.
<b>LogPanelFilename</b>	String	Establece el nombre completo del archivo donde se almacena la actividad del panel. Si se deja vacío, se utiliza el archivo PanelLog.yyyyMMdd.txt.
<b>MinimizeDoEventsCalls</b>	Boolean	Establece que se reduzcan a un mínimo las llamadas a DoEvents dentro del componente. Esto reduce la respuesta de la aplicación pero permite una ejecución más rápida del componente.
<b>MonitorMode</b>	MonitorModes	Establece el criterio utilizado para hacer parpadear los leds de la interfaz (TxRx, StartFinish, AlwaysOn, AlwaysOff).
<b>OnErrorReconnect</b>	Boolean	Establece si ante una caída de la conexión, el componente debe intentar reconectarse automáticamente.
<b>PanelFont</b>	Font	Establece el fuente del texto utilizado en el panel.
<b>PanelLines</b>	Int16	Establece la cantidad de líneas que almacena el panel.
<b>PanelMode</b>	PanelModes	Establece cómo se muestran los telegramas en el panel (Hexadecimal, ASCII, Decimal).
<b>PanelText</b>	String	Devuelve el texto presente en el panel.
<b>PanelWordWrap</b>	Boolean	Establece la propiedad WordWrap del panel.
<b>PCId</b>	String	Devuelve el PCId de la máquina donde se está ejecutando el componente.
<b>ShowConnectionStatus</b>	Boolean	Establece si se muestra el estado de la conexión en la interfaz.
<b>ShowElapsedTimeBar</b>	Boolean	Establece si se muestra la barra de progreso de tiempo en la interfaz.
<b>ShowPanel</b>	Boolean	Establece si se muestra el panel en la interfaz.
<b>ShowToolTips</b>	Boolean	Establece si se muestran tooltips descolgables en la interfaz.
<b>SilentMode</b>	Boolean	Establece si la interfaz debe trabajar en modo silencioso (no se actualiza la interfaz ni el panel).
<b>Status</b>	String	Muestra un mensaje con el último estado del componente, que combina el último estado del driver o la última excepción, según cuál de los dos se haya producido último.
<b>StatusBarFont</b>	Font	Establece el fuente del texto utilizado en la barra del menú.
<b>TcpIpAddress</b>	String	Dirección IP utilizada en la última conexión tcp/ip abierta.
<b>TcpIpHiddenAddress</b>	String	Si no está vacía, es la dirección IP que será realmente utilizada en las llamadas Connect y Ping. La IP que

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

		se mostrará en los mensajes del panel de comunicaciones seguirá siendo la establecida en TcplpAddress o la suministrada al llamar a los métodos. Esta IP oculta se utiliza para no mostrar la verdadera IP a la que se está realizando una comunicación.
<b>TcplpPortNum</b>	Int32	Puerto IP utilizado en la última conexión tcp/ip abierta.
<b>TcplpProtocol</b>	TcplpProtocolType	Protocolo IP utilizado en la última conexión tcp/ip abierta.
<b>TcplpT3Delay</b>	Int32	Retardo T3 utilizado en la última conexión tcp/ip abierta.
<b>TcplpTimeOut</b>	Int32	Puerto IP utilizado en la última conexión tcp/ip abierta.
<b>Text</b>	String	Establece el texto que aparecerá como título en el marco que rodea al componente.
<b>TextFont</b>	Font	Establece el fuente del texto del marco que rodea al componente.

## Métodos

Nombre del método	Valor retornado	Descripción
<b>AbortCommunication</b>	Boolean	Termina con una comunicación que se está llevando a cabo, sin esperar a que termine normalmente o a que se agote el tiempo de espera establecido.
<b>AsyncPing</b>	Boolean	Realiza un Ping a una dirección IP de manera asincrónica, es decir, sin bloquear completamente la aplicación (permite que el formulario de la aplicación siga respondiendo).
<b>AsyncReadAnalog</b>	Boolean	Lee valores analógicos de manera asincrónica (sin bloqueo).
<b>AsyncReadBinary</b>	Boolean	Lee valores binarios de manera asincrónica (sin bloqueo).
<b>AsyncWriteAnalog</b>	Boolean	Escribe valores analógicos de manera asincrónica (sin bloqueo).
<b>AsyncWriteBinary</b>	Boolean	Escribe valores binarios de manera asincrónica (sin bloqueo).
<b>ClearPanel</b>	void	Limpia el contenido del panel.
<b>Connect</b>	Boolean	Unificación de TcplpConnect y SerialConnect. Establece una conexión serie o tcp/ip.
<b>Disconnect</b>	Boolean	Fuerza la desconexión de una conexión abierta.
<b>GetConnectionParameters</b>	String	Devuelve un texto descriptivo con los parámetros utilizados en la última conexión.
<b>GetDriverCheckLicense</b>	String	Averigua si un driver requiere de una licencia.
<b>GetDriverDescription</b>	String	Obtiene la descripción de un driver por su nombre de archivo.
<b>GetDriverDeveloper</b>	String	Obtiene el nombre del desarrollador de un driver por su nombre de archivo.
<b>GetDriverLicenseType</b>	String	Obtiene el tipo de licencia de un driver por su nombre de archivo.
<b>GetDriverParameters</b>	String	Devuelve un texto descriptivo con los

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

		parámetros utilizados en la última lectura o escritura.
<b>GetDriverVersion</b>	String	Obtiene la versión de un driver por su nombre de archivo.
<b>GetLastPointValue</b>	double	Devuelve el valor anterior de un PointValue.
<b>GetMachineKey</b>	String	Obtiene el PCId de la computadora en la que se ejecuta la aplicación.
<b>GetPointDate</b>	DateTime	Obtiene la fecha del último PointValue.
<b>GetPointTagName</b>	String	Obtiene un nombre de tag asociado a un PointValue que es previamente suministrado por el usuario.
<b>GetPointText</b>	String	Obtiene un texto asociado a un PointValue que puede haber sido devuelto por el driver acompañando el valor numérico.
<b>GetPointValue</b>	Double	Obtiene el valor correspondiente a un PointValue tal como fue recibido desde un equipo.
<b>GetPointValueChanged</b>	Boolean	Averigua si un PointValue cambió su valor en la última lectura respecto de la anterior.
<b>GetRunningBitMode</b>	String	Devuelve "32" si la aplicación se está ejecutando en 32 bits, y "64" si se está ejecutando en 64 bits.
<b>InjectSimulatedReply</b>	void	Inyecta un telegrama (escrito en hexadecimal por ejemplo "010AFE9B") como respuesta a un driver, de manera tal de simular una respuesta de un equipo real. El telegrama debe inyectarse <u>antes</u> de realizar la llamada al método de lectura o escritura, para que esté disponible. El telegrama puede ser suministrado hasta en dos partes.
<b>LastPointValue</b>	double/ Boolean	Sobrecarga de GetLastPointValue y SetLastPointValue.
<b>LogPanelHeader</b>	void	Agrega al log de actividad del panel un encabezado con información del driver, fecha y hora y datos de la conexión y del comando utilizado.
<b>Ping</b>	Boolean	Realiza un Ping a una dirección IP.
<b>PointDate</b>	DateTime/Boolean	Unificación de GetPointDate y SetPointDate.
<b>PointTagName</b>	String/Boolean	Unificación de GetPointTagName y SetPointTagName.
<b>PointText</b>	String/Boolean	Unificación de GetPointText y SetPointText.
<b>PointValue</b>	Double/Boolean	Unificación de GetPointValue y SetPointValue.
<b>PointValueChanged</b>	Boolean	Unificación de GetPointValueChanged y SetPointValueChanged.
<b>ReadAnalog</b>	Boolean	Unificación de AsyncReadAnalog y SyncReadAnalog.
<b>ReadBinary</b>	Boolean	Unificación de AsyncReadBinary y SyncReadBinary.
<b>ReadBooleanValues</b>	Boolean	Idem ReadBinary.
<b>ReadNumericValues</b>	Boolean	Idem ReadAnalog.
<b>Reconnect</b>	Boolean	Intenta restablecer una conexión utilizando los mismos parámetros de

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

<b>ResetLastDriverStatus</b>	void	la conexión anterior. Resetea la propiedad LastDriverStatus a un texto vacío.
<b>ResetLastException</b>	void	Resetea la propiedad LastException a un texto vacío.
<b>SavePanel</b>	Boolean	Guarda el contenido del panel en el archivo que se indique.
<b>SerialConnect</b>	Boolean	Establece una conexión mediante un puerto serial.
<b>SetLastPointValue</b>	Boolean	Redefine el valor anterior de un PointValue.
<b>SetPointDate</b>	Boolean	Redefine la fecha del último PointValue.
<b>SetPointTagName</b>	Boolean	Establece un nombre de tag asociado a un PointValue.
<b>SetPointText</b>	Boolean	Establece un texto asociado a un PointValue, que puede ser requerido por un driver para su envío.
<b>SetPointValue</b>	Boolean	Establece el valor correspondiente a un PointValue para ser enviado a un equipo.
<b>SetPointValueChanged</b>	Boolean	Sobre-escribe la condición que indica si un PointValue cambió su valor en la última lectura respecto de la anterior.
<b>SyncPing</b>	Boolean	Realiza un Ping a una dirección IP, que bloquea la aplicación hasta que se termina de ejecutar el comando.
<b>SyncReadAnalog</b>	Boolean	Lee valores analógicos de manera sincrónica (con bloqueo).
<b>SyncReadBinary</b>	Boolean	Lee valores binarios de manera sincrónica (con bloqueo).
<b>SyncWriteAnalog</b>	Boolean	Escribe valores analógicos de manera sincrónica (con bloqueo).
<b>SyncWriteBinary</b>	Boolean	Escribe valores binarios de manera sincrónica (con bloqueo).
<b>TcplpConnect</b>	Boolean	Establece una conexión mediante un vínculo tcp/ip.
<b>UnloadDriver</b>	void	Descarga el driver de la memoria, de manera que el archivo .tlk no quede retenido por el sistema operativo.
<b>WriteAnalog</b>	Boolean	Unificación de AsyncWriteAnalog y SyncWriteAnalog.
<b>WriteBinary</b>	Boolean	Unificación de AsyncWriteBinary y SyncWriteBinary.
<b>WriteBooleanValues</b>	Boolean	Idem WriteBinary.
<b>WriteNumericValues</b>	Boolean	Idem WriteAnalog.
<b>WriteToPanel</b>	void	Agrega un texto de usuario al panel.

(Los parámetros requeridos por cada método con sus sobrecargas se pueden consultar desde el entorno de desarrollo con la funcionalidad IntelliSense del IDE, también llamada "autocompletar", que consiste en desplegar información de ayuda al usuario a medida que se escribe el código).

## Eventos

Nombre del evento	Argumentos	Descripción
<b>Connected</b>	autoreconnected	Informa que se estableció una nueva conexión, indicando además si la misma fue producto de una reconexión automática.
<b>Disconnected</b>		Informa que una conexión se cerró.
<b>Finished</b>	finishedok	Informa que una transacción de

CPKSoft Ingeniería

Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

<b>NewPanelMessage</b>	message	lectura o escritura finalizó, indicando si lo hizo de manera exitosa o fallida. Informa que se agregó un nuevo mensaje al panel y pasa el texto como argumento.
<b>User</b>		Evento de usuario disparado cuando se hace click sobre el indicador del estado de la conexión.

CPKSoft Ingeniería  
Drivers para comunicación industrial.

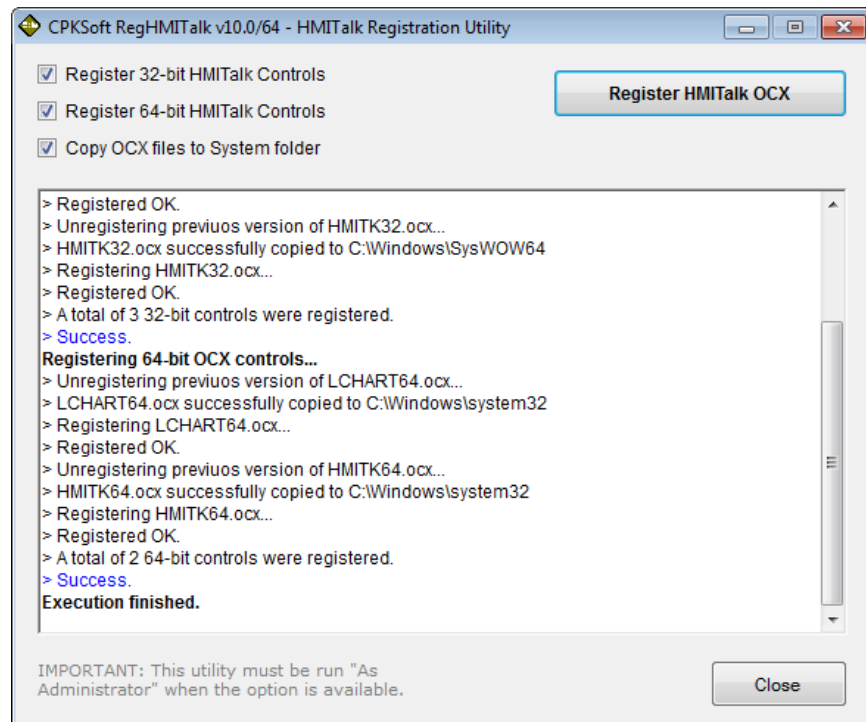
[www.cpksoft.com.ar](http://www.cpksoft.com.ar)  
[www.facebook.com/cpksoftingenieria](https://www.facebook.com/cpksoftingenieria)  
[cpksoftingenieria@hotmail.com](mailto:cpksoftingenieria@hotmail.com)  
tel: 54-11-1545788354

1990-2012



# Soluciones de comunicación industrial para Windows

ello se provee la herramienta **RegHMITalk.exe**, que invoca al utilitario regsvr32.exe de Windows que corresponda para registrar correctamente a HMITalk y sus dependencias.



Antes de correr RegHMITalk, y sobre todo si utiliza sistemas operativos anteriores a Windows 7, debe asegurarse de contar con todas las dll de Windows requeridas para ejecutar aplicaciones MFC/C++. Si tiene XP es necesario tener la versión SP2 instalada, o bien colocar manualmente los archivos mfc100.dll y msucr100.dll en su máquina, sino recibirá un error al registrar o utilizar los objetos. Para asegurarse de contar con todas las librerías de Microsoft necesarias en su sistema, puede descargar el paquete redistribuible "Microsoft Visual C++ 2010 Redistributable Package (x86/x64)" desde la página de Microsoft, llamado vc\_redist\_x86/x64.exe, quien se ocupa de instalar todos los archivos necesarios. Este paso se debe hacer en todas las máquinas que presenten problemas al registrar los ocx.

Cuando el sistema operativo sea de 64 bits, existen dos versiones diferentes de regsvr32.exe, una para 32 bits (en c:\windows\sysWOW64) y otra para 64 bits (en c:\windows\system32), y RegHMITalk se ocupa en cada caso de invocar la versión correcta.

Se recomienda registrar ambas versiones de 32 y 64 bits siempre que su máquina y sistema operativo lo permitan, especialmente si usted planea construir aplicaciones que deban ejecutarse luego como programas de 64 bits.

Además, se sugiere copiar todos los .ocx en la carpeta c:\Windows\system32 y también en la carpeta c:\Windows\SysWOW64 (si ésta existe en su sistema operativo), ya que las interfaces de desarrollo pueden ir a buscar allí los ocx en tiempo de diseño.

Por otra parte, un problema conocido con las plataformas Visual Studio que trabajan en 32 bits es que sólo trabajan bien en diseño si la CPU de destino establecida para la aplicación es de 32 bits. Si se compila una aplicación para CPU de destino de 64 bits, una vez compilada se recomienda volver a establecer la plataforma de destino en 32 bits. La aplicación generada (el archivo ejecutable) funcionará correctamente en 64 bits. Pero si se omite volver a establecer el destino como de 32 bits y guardamos el proyecto apuntando a una CPU de 64 bits y luego cerramos Visual Studio, al ingresar nuevamente al proyecto nos dará un error al mostrar los formularios que contengan los objetos ocx.

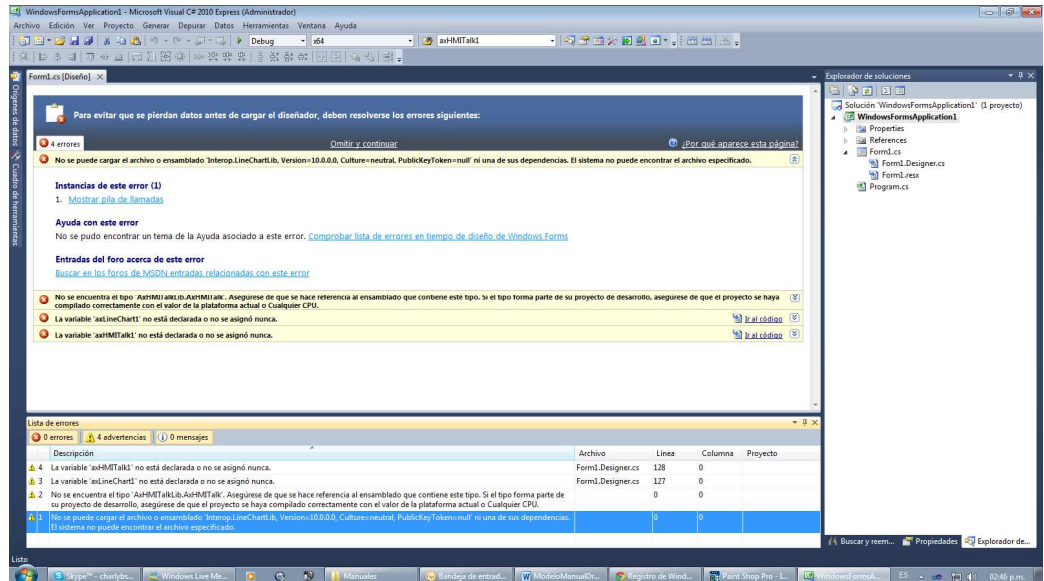
CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

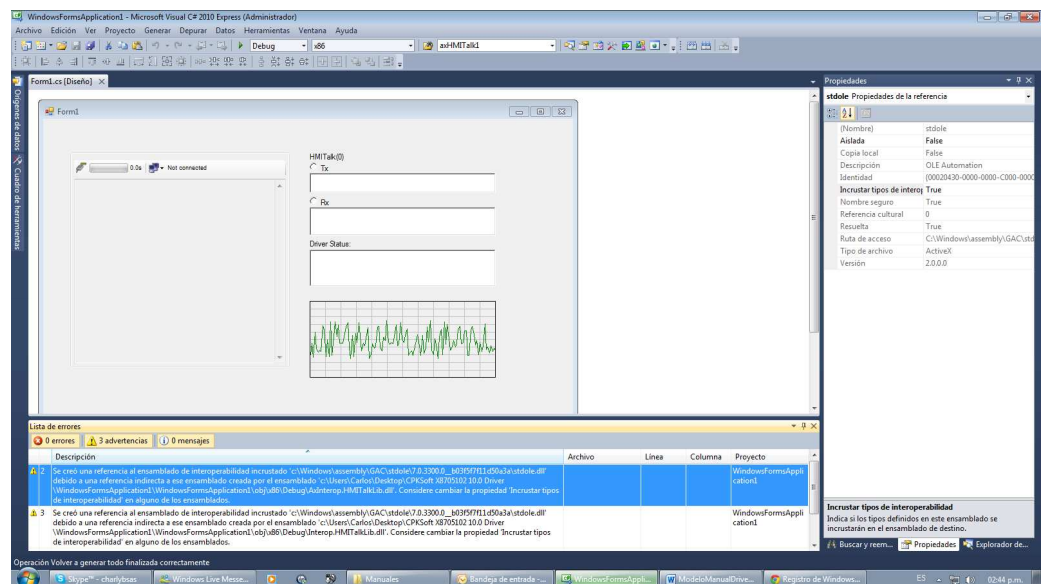


# Soluciones de comunicación industrial para Windows



El problema se resuelve estableciendo la plataforma de destino nuevamente a 32 bits (mientras persiste el error), guardando el proyecto de la aplicación, cerrando Visual Studio y volviendo a cargar el proyecto. Este error se produce porque Visual Studio se ejecuta siempre en 32 bits, y si la plataforma quedó establecida en 64 bits, al recargar el proyecto trata de utilizar las versiones de 64 bits de los ocx, en sus formularios, lo que causa un conflicto de plataforma.

Otro problema conocido con aplicaciones que utilizan ocx desarrollados en MFC/C++ es que dependen del archivo stdole.dll que es parte de Windows y al compilar pueden aparecer advertencias del tipo: "Se creó una referencia al ensamblado de interoperabilidad incrustado 'c:\Windows\assembly\GAC\stdole\7.0.3300.0\_b035f7f11d50a3a\stdole.dll' debido a una referencia indirecta a ese ensamblado creada por el ensamblado 'c:\...\AxInterop.HMTalkLib.dll'. Considere cambiar la propiedad 'Incrustar tipos de interoperabilidad' en alguno de los ensamblados".



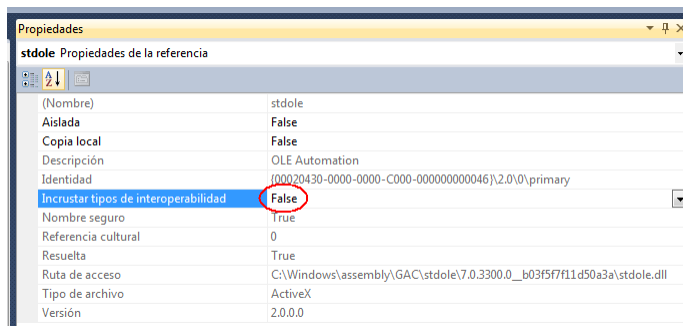
Si esta advertencia aparece, se debe ir a las referencias del proyecto, localizar el archivo stdole32.dll, ir a sus propiedades y allí marcar "Incrustar tipos de interoperabilidad" en false. Al compilar nuevamente, desaparece el error.

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows



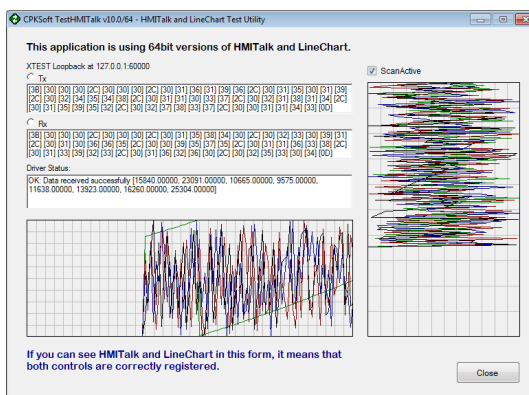
Finalmente, es muy importante tener en cuenta que tanto si se utiliza Visual Basic 6.0 (VB6) como Visual Studio C#.NET, Visual Basic.NET o cualquier otro entorno de desarrollo desde el que se tengan que agregar los componentes a una barra de herramientas en tiempo de diseño, es necesario siempre ejecutar estos entornos de desarrollo **en modo administrador**, cuando el sistema operativo lo permita (como es el caso de Windows Vista y Windows 7 por ejemplo) para que no se produzca un error al

## Resumen:

1. Instalar todas las dll requeridas para utilizar ocx en su sistema. Si es necesario, instalar el paquete redistribuible vc\_redist\_x86/x64.exe para MFC/C++ 2010 desde la página de Microsoft.
2. Correr el programa RegHMITalk.exe en modo administrador (si el sistema le da la opción), tanto para 32 bits como para 64 bits (éste último si la CPU de la máquina lo permite).
3. Correr el entorno de desarrollo que utilice en modo administrador (si el sistema le da la opción).
4. Copiar todos los ocx que forman parte del driver en las carpetas c:\windows\system32 y en c:\windows\syswow64 (cuando esta última esté disponible). Puede valerse de la herramienta RegHMITalk para asegurar este requisito.
5. **Cuando se compilen aplicaciones para CPU de 64 bits en Visual Studio, volver a apuntar la CPU en el compilador a 32 bits antes de guardar el proyecto, para que no falle la próxima vez que se abra.**
6. Si al compilar en Visual Studio aparece una advertencia relacionada con stdole.dll, ir a las Referencias del proyecto y en las propiedades de stdole.dll establecer "Incrustar tipos de interoperabilidad" en False.

## Prueba de funcionamiento

Para asegurarse de que los componentes ActiveX/OCX se han registrado correctamente en su sistema, se sugiere correr el utilitario TestHMITalk.exe. Si al ejecutarlo le aparece un formulario donde se muestra los objetos HMITalk y LineChart en funcionamiento, esto indica que no han existido conflictos.



CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

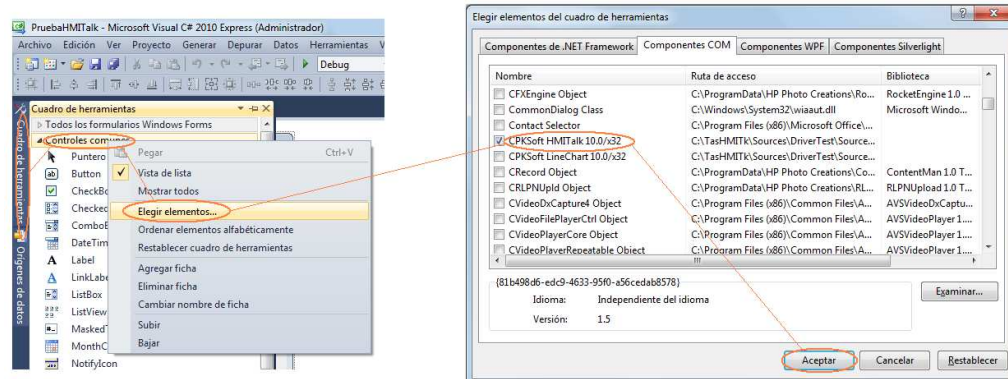
# Soluciones de comunicación industrial para Windows

## Crear una aplicación

Para mostrar cómo crear una aplicación utilizaremos Microsoft Visual C# 2010 Express.

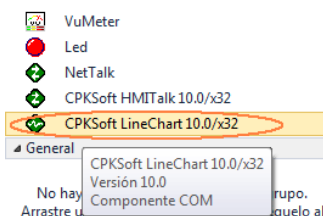
Para poder agregar el objeto HMITalk debemos antes haberlo registrado con RegHMITalk.exe (para 32 bits y si se puede, también para 64 bits), para que aparezca entre los componentes COM disponibles.

Suponiendo que HMITalk ya fue correctamente registrado, hay que para agregarlo al cuadro de herramientas de Visual Studio. Para ello deberemos posicionarnos sobre el *Cuadro de herramientas* y sobre *Controles comunes* presionar botón derecho del mouse donde aparecerá un menú descolgable. Allí seleccionamos la opción *Elegir elementos...* y luego seleccionamos la pestaña *Componentes COM*. Buscamos el componente CPKSoft HMITalk en la lista, lo marcamos y aceptamos:



Vale hacer aquí una aclaración importante a ser tenida en cuenta: si usted está trabajando con una máquina de 64 bits, con sistema operativo de 64 bits y necesita crear una aplicación de 64 bits que utilice HMITalk, una vez que haya registrado el objeto (tanto para 32 bits como para 64 bits), lo esperable será que seleccione la versión de 64 bits de HMITalk desde el listado de Componentes COM disponibles en su Visual Studio 2010 o la plataforma de desarrollo que utilice. Pero lo que realmente sucederá probablemente será que los componentes que Visual Studio le va a mostrar en la lista van a depender de si el propio Visual Studio corre en 32 o en 64 bits. El Visual C#.NET 2010 Express gratuito que utilizamos en nuestros ejemplos está escrito por Microsoft como una aplicación de 32 bits (a pesar de que es capaz de producir aplicaciones de 64 bits sin problemas), por lo que la versión del componente HMITalk que le aparecerá en el listado será solamente la de 32 bits. En realidad sólo le va a mostrar el componente de 64 bits cuando el propio Visual Studio está corriendo en 64 bits, lo que probablemente sucederá con versiones futuras de ese entorno de desarrollo. No obstante, mientras tanto esto no significa que no se puedan desarrollar aplicaciones de 64 bits con HMITalk desde un Visual Studio de 32 bits. Simplemente seleccione el HMITalk de 32 bits que allí se ofrece, y utilícelo normalmente. Sepa que mientras se esté en modo diseño, donde Visual Studio de 32 bits es quien maneja el componente, la versión de HMITalk que se estará mostrando en sus formularios será la de 32 bits. Cuando se ejecute la aplicación desarrollada por usted para 64 bits, ya sea desde dentro o desde fuera del IDE, la versión de HMITalk que estará allí corriendo será la de 64 bits. Por eso es importante que estén registradas ambas versiones de HMITalk, ya que Windows pasa automáticamente de una a otra dependiendo de si el entorno de ejecución sea de 32 o 64 bits, sin que usted lo note. También es importante que cuando se distribuya la aplicación, se considere incluir los dos archivos HMITK32.ocx y HMITK64.ocx para asegurar la compatibilidad en ambos casos, junto con ambas versiones de los drivers utilizados (\*.32.tlk y \*.64.tlk).

Una vez marcado el componente HMITalk de la lista, le damos aceptar y nos aparecerá el componente en el cuadro de herramientas, listo para ser arrastrado a nuestro formulario.



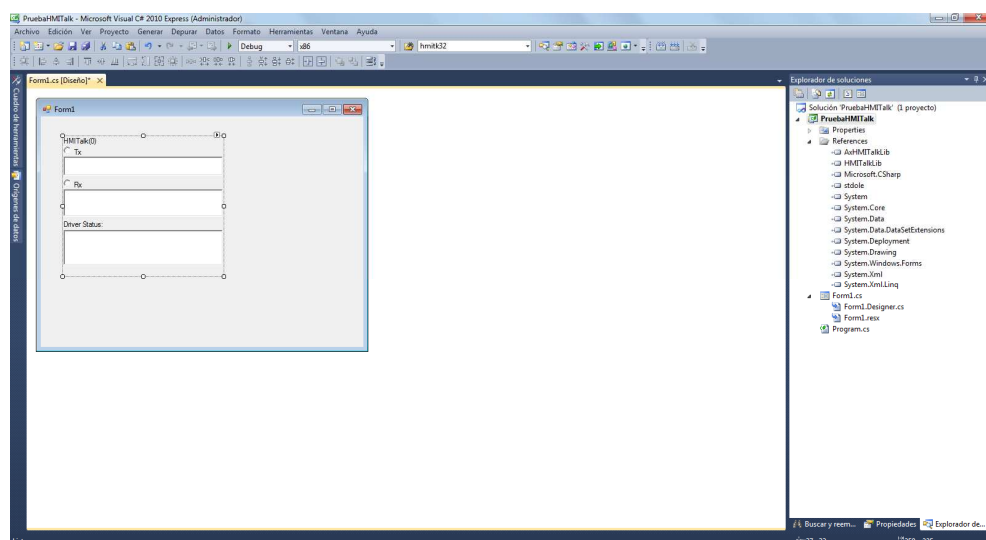
CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

En la imagen siguiente se ve el componente HMITalk ya colocado en el formulario, cuyo tamaño se ha agrandado para recibir al objeto:



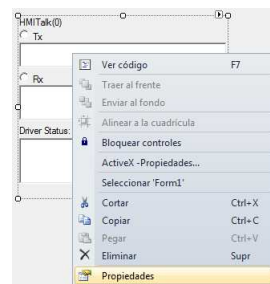
En este punto, usted puede hacer una prueba de compilación del proyecto, que por supuesto aún no hace nada más que mostrar el objeto en pantalla. Si le llegara a aparecer un error de este tipo: "Advertencia: Se creó una referencia al ensamblado de interoperabilidad incrustado 'c:\...\stdole.dll' debido a una referencia indirecta a ese ensamblado creada por el ensamblado 'c:\...\AxInterop.HMITalkLib.dll'. Considere cambiar la propiedad 'Incrustar tipos de interoperabilidad' en alguno de los ensamblados", la solución es simple y consiste en lo siguiente: en el Explorador de Soluciones, en las Referencias del proyecto, localice el archivo "stdole", vaya a sus propiedades y desactive la casilla "Incrustar tipos de interoperabilidad". Otra alternativa es directamente eliminar stdole de sus Referencias. Compile nuevamente su proyecto y compruebe que el problema se solucionó.

A continuación se explican las acciones más comunes del objeto HMITalk, como es el caso de la configuración del puerto y el manejo de lecturas y escrituras.

## Configuración del puerto de comunicaciones

En general el objeto HMITalk se puede configurar totalmente en tiempo de diseño a través de sus páginas de propiedades.

Para configurar el puerto de comunicaciones, se debe ir a las pestañas TCP/IP y Comm, que se acceden con botón derecho sobre el objeto, y seleccionando luego la opción "Propiedades".



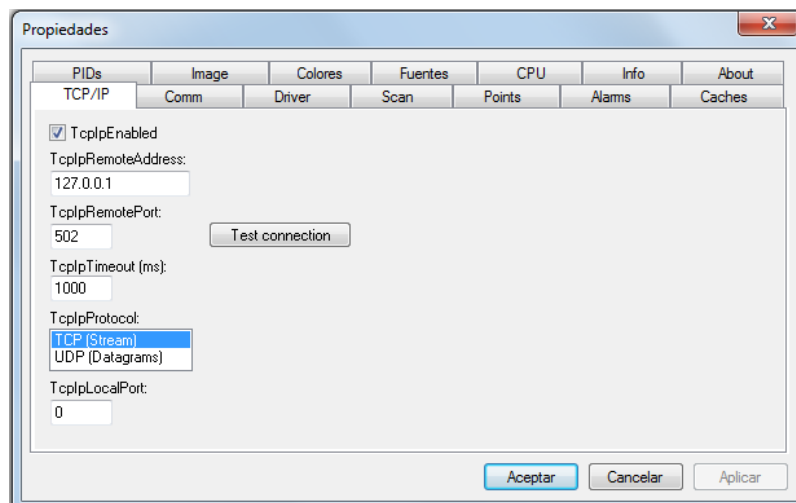
La pestaña TCP/IP permite establecer si la comunicación se va a realizar sobre TCP/IP. De ser así, se debe activar la casilla TcplpEnabled y establecer los demás parámetros de la comunicación TCP/IP:

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

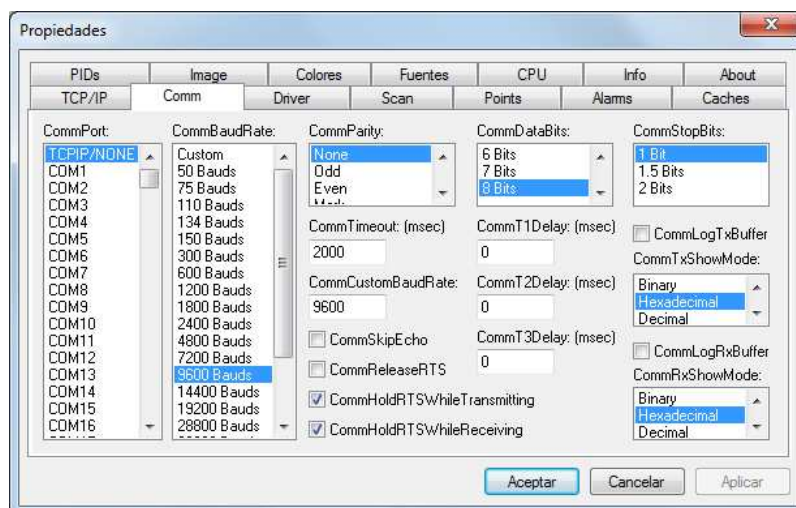
www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows



La pestaña Comm permite establecer las propiedades del puerto serial, en el caso que la comunicación no se realice por TCP/IP sino por un puerto COM serial:



Se pueden colocar varias instancias de objetos compartiendo un mismo puerto de comunicaciones, tanto serial como TCP/IP. El motor manejador de los objetos que corre detrás de la escena se encarga de ir dándole permiso de uso del vínculo a cada objeto que así lo requiera.

## Manejo de lecturas

Una vez configurado el puerto de comunicaciones, se debe seleccionar el driver que se va a utilizar y configurar sus parámetros. Esto se realiza en la pestaña Drivers de las páginas de propiedades que se muestra a continuación, y que ya está configurada para producir la misma lectura que se utilizó como ejemplo para el componente NetTalk:

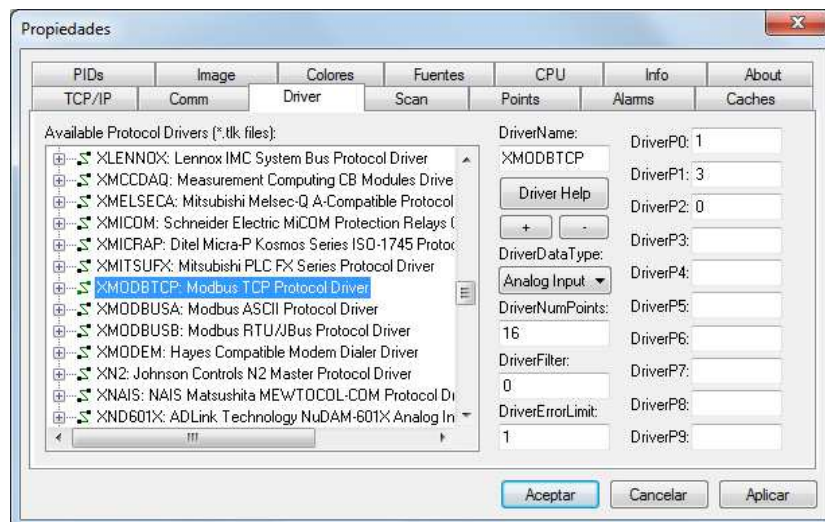
CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012



# Soluciones de comunicación industrial para Windows



Obsérvese que, a diferencia del componente NetTalk, en este caso no es necesario escribir ninguna línea de código para configurar el objeto.

Para HMITalk es válido todo lo ya explicado para el componente NetTalk en cuando a cómo configurar las propiedades o parámetros relacionados con el comando. Se utilizan los mismos parámetros DriverP0 a DriverP9, con idénticos significados.

La propiedad DriverDataType de HMITalk es el modo de establecer el tipo de operación que se va a realizar. Las equivalencias con NetTalk son las siguientes:

- Analog Input equivale a ReadNumericValues
- Digital Input equivale a ReadBooleanValues
- Analog Output equivale a WriteNumericValues
- Digital Output equivale a WriteBooleanValues

## Lecturas automáticas por eventos

Para activar las comunicaciones automáticas, se debe acceder a la pestaña Scan, donde se debe activar la propiedad ScanActive para que el motor de comunicaciones realice un puleo automático mientras la aplicación se esté ejecutando.

Una aplicación típica puede consistir de varios objetos HMITalk corriendo en simultáneo, cada uno configurado para apuntar a diferentes equipos o a diferentes porciones de memoria o de datos de un mismo equipo.

Cuando se crea el primer objeto en la aplicación, éste a su vez inicializa un motor de puleo interno que trabaja en background y que lleva la cuenta de todos los demás objetos que corren en la misma aplicación, manejando los accesos a los diferentes puertos y otorgándoles las prioridades necesarias para hacer uso de los mismos cuando se les cumple el tiempo de realizar cada comunicación.

El usuario no necesita colocar un objeto Timer en su aplicación para resolver el barrido automático ya que de esto se encarga el propio HMITalk.

CPKSoft Ingeniería

Drivers para comunicación industrial.

www.cpksoft.com.ar

www.facebook.com/

cpksoftingenieria

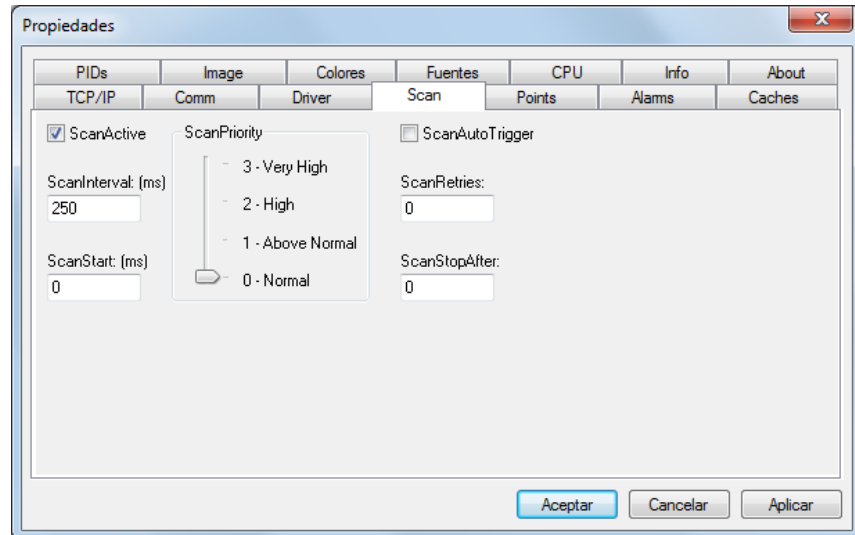
cpksoftingenieria@

hotmail.com

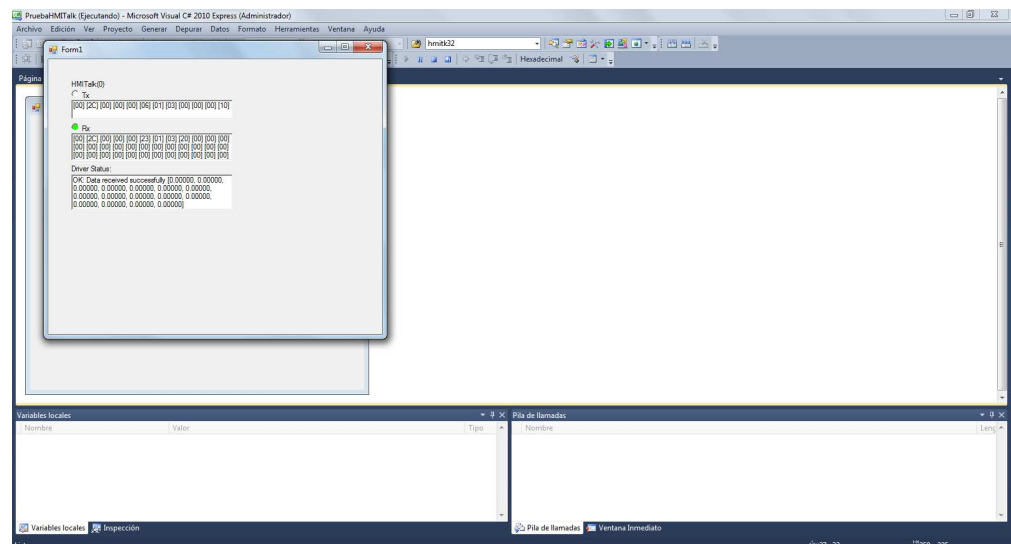
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows



En tiempo de ejecución, se podrá ver al objeto comunicándose, mostrando los leds parpadeando y los telegramas intercambiados en las cajas de texto de su interfaz:



Cuando se deja la propiedad ScanActive activada, el motor polea periódicamente cada un cierto tiempo que se establece en la propiedad ScanInterval.

La forma de obtener los datos recibidos es típicamente mediante el evento OnPointValueChanged, que se produce cada vez que alguno de los datos recibidos ha cambiado su valor con respecto a su valor anterior.

Por ejemplo, si se quisieran tomar los tres primeros valores recibidos y asignarlos a tres objetos Label diferentes (label1, label2 y label3), el código sería así:

```
private void axHMITalk1_OnPointValueChanged(object sender,
AxHMITalkLib._DHMITalkEvents_OnPointValueChangedEvent e)
{
    switch (e.point)
    {
```

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012



# Soluciones de comunicación industrial para Windows

```
        case 0: this.label1.Text =
this.axHMITalk1.get_PointValue(e.point).ToString(); break;
        case 1: this.label2.Text =
this.axHMITalk1.get_PointValue(e.point).ToString(); break;
        case 2: this.label3.Text =
this.axHMITalk1.get_PointValue(e.point).ToString(); break;
    }
}
```

La particularidad de este evento es que, al ejecutarse ante el cambio en un determinado punto, si cambian tres puntos, el evento se dispara tres veces. La diferencia entre cada llamada es que la propiedad e.Point va cambiando para apuntar al índice del punto que cambió. Por ello se utiliza una sentencia switch o select que permita actualizar el objeto que corresponda de acuerdo a qué dato cambió.

Otra manera de recibir los valores, ignorando si hubo o no hubo cambios, es utilizando el evento OnSuccessfullyReceived. Este evento se dispara al final de la comunicación, siempre y cuando la misma haya sido exitosa. En este caso, el código podría quedar así:

```
private void axHMITalk1_OnSuccessfullyReceived(object sender, EventArgs e)
{
    this.label1.Text = this.axHMITalk1.get_PointValue(0).ToString();
    this.label2.Text = this.axHMITalk1.get_PointValue(1).ToString();
    this.label3.Text = this.axHMITalk1.get_PointValue(2).ToString();
}
```

Si se produjo un error en la comunicación, lo habitual es escuchar el evento OnErrorReceiving y consultar el mensaje de error en la propiedad DriverStatus, por ejemplo así:

```
private void axHMITalk1_OnErrorReceiving(object sender, EventArgs e)
{
    MessageBox.Show(this.axHMITalk1.DriverStatus);
}
```

Otra manera de reportar un error es directamente escuchar el evento OnDriverStatusChanged, que reporta cada vez que DriverStatus pasó del valor que tenía a un valor diferente. De esta manera, si persiste un mismo error, este evento se dispara una única vez cuando se produjo el error.

Existen varios otros eventos relacionados con el manejo de la comunicación que se pueden ver en el listado de eventos más adelante y su uso se puede inducir por lo general del propio nombre del evento.

## Lecturas a demanda por eventos

Las lecturas a demanda son un caso especial donde en lugar de mantener la propiedad ScanActive activada, se deja que la aplicación sea quien solicite una lectura mediante el llamado al método Trigger() del objeto de la siguiente manera:

```
private void button1_Click(object sender, EventArgs e)
{
    this.axHMITalk1.Trigger();
}
```

Los eventos que se disparan son exactamente los mismos que en una lectura automática. Incluso la propiedad ScanActive se puede activar en tiempo de ejecución, pasando de lectura automática a manual según lo requiera la aplicación.

## Lecturas con espera

Las lecturas con espera son un caso poco utilizado con HMITalk ya que es un objeto que está diseñado para reportar todas las situaciones posibles mediante el disparo de diferentes eventos. No obstante, es

# Soluciones de comunicación industrial para Windows

posible iniciar una lectura que bloquee la ejecución de la aplicación, a la espera de que una comunicación finalice antes de continuar con la ejecución de la siguiente línea de código.

Para ello se utiliza el método `TriggerAndWait`, quien devuelve `true` o `false` de acuerdo a si la comunicación fue exitosa o si terminó con errores. Por ejemplo, combinando lo que ya hacían los ejemplos anteriores, se tendría este código:

```
private void button2_Click(object sender, EventArgs e)
{
    if (this.axHMITalk1.TriggerAndWait())
    {
        this.label1.Text = this.axHMITalk1.get_PointValue(0).ToString();
        this.label2.Text = this.axHMITalk1.get_PointValue(1).ToString();
        this.label3.Text = this.axHMITalk1.get_PointValue(2).ToString();
    }
    else MessageBox.Show(this.axHMITalk1.DriverStatus);
}
```

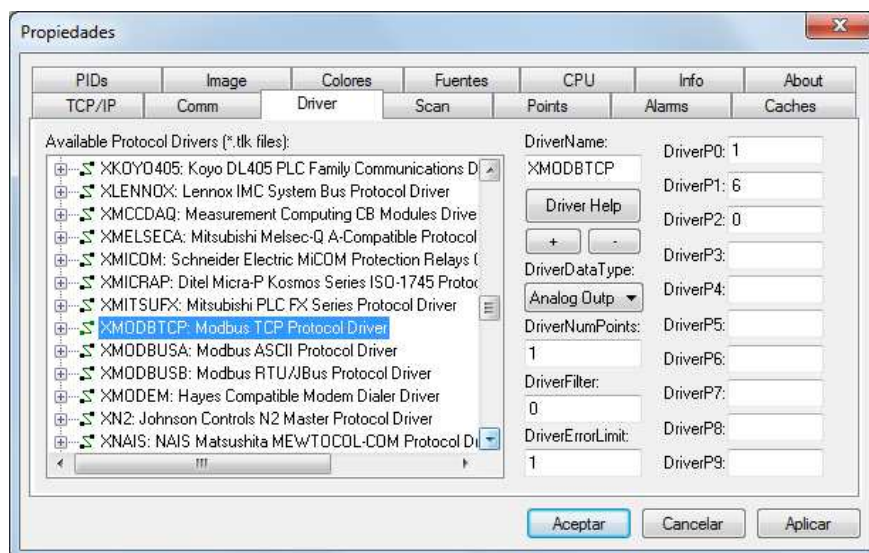
Es importante señalar que, aunque haya espera, al terminar la comunicación igual se disparan todos los eventos relacionados con la aplicación si éstos están siendo escuchados.

## Manejo de escrituras

El manejo típico de las escrituras desde HMITalk es básicamente una acción que se realiza a demanda donde se envían uno o más datos a un equipo por única vez. Lo normal es crear un objeto nuevo que esté dedicado solamente a manejar escrituras, o bien crear varios objetos, si se requiere enviar escrituras simultáneamente a través de varios puertos o a varios equipos.

Es raro mantener la propiedad `ScanActive` encendida en los objetos encargados de escrituras, ya que esto causaría que se estén reenviando siempre los mismos datos a un equipo. Puede ser útil si dichos datos cambian todo el tiempo, como podría ser el caso de enviar una señal de control o bien mantener actualizado un watchdog para que un equipo sepa que la aplicación permanece activa, pero normalmente el scan se deja apagado.

En la imagen siguiente se muestra cómo configurar las propiedades del driver de un segundo objeto para realizar la misma escritura que se ejemplificó para NetTalk:



CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

El procedimiento habitual consiste en cargar el valor a ser enviado y luego invocar al método Trigger, como en el caso de las lecturas a demanda. Un ejemplo de esto es el siguiente:

```
private void button3_Click(object sender, EventArgs e)
{
    this.axHMITalk2.set_PointValue(0, 100.0);
    this.axHMITalk2.Trigger();
}
```

En caso que se haya producido un error al hacer el envío del dato, el objeto disparará el evento OnErrorSending. Al igual que con el evento OnErrorReceiving (que se dispara ante un error al recibir datos), aquí también se puede consultar el valor de la propiedad DriverStatus para obtener los detalles del error:

```
private void axHMITalk2_OnErrorSending(object sender, EventArgs e)
{
    MessageBox.Show(this.axHMITalk2.DriverStatus);
}
```

No es usual usar el método TriggerAndWait() porque por lo general el envío de datos o escrituras se realiza a demanda y una vez despachadas, solo interesa saber si se produjo algún tipo de error. Pero si fuera necesario asegurarse de que el envío se realizó correctamente antes de continuar con alguna otra acción, puede ser útil utilizar la variante con espera de ejecución.

## Guía rápida de propiedades, métodos y eventos

### Propiedades

AlarmsInfo	ArrayOfPointValueDates	ArrayOfPointValues
ArrayOfValuesFormat	CachesInfo	Caption
CommBaudRate	CommCustomBaudRate	CommDataBits
CommHoldRTSWhileReceiving	CommHoldRTSWhileTransmitting	CommLogRxBuffer
CommLogTxBuffer	CommParity	CommPort
CommReleaseRTS	CommRxBuffer	CommRxShowMode
CommSkipEcho	CommStopBits	CommT1Delay
CommT2Delay	CommT3Delay	CommTimeout
CommTxBuffer	CommTxShowMode	CommunicationAborted
CPUConsumption	CPUPriority	DriverDataType
DriverEfficiency	DriverErrorCount	DriverErrorLimit
DriverErrorPresent	DriverFilter	DriverName
DriverNumPoints	DriverP0	DriverP1
DriverP2	DriverP3	DriverP4
DriverP5	DriverP6	DriverP7
DriverP8	DriverP9	DriverStatus
ElapsedSeconds	EnableAlarms	EnableCaches
EnableCalculations	EnablePIDs	EnableTimer
ImgAutoSize	ImgBackColor	ImgBoxAppearance
ImgFontAppearance	ImgForeColor	ImgHideAll
ImgInvertTxRx	ImgLedAppearance	ImgRxBoxBackColor
ImgRxBoxLines	ImgRxCaption	ImgRxLedBackColor
ImgRxLedColor	ImgRxShowDetails	ImgRxShowLed
ImgStatusBoxBackColor	ImgStatusBoxLines	ImgStatusCaption
ImgStatusShowDetails	ImgTxBoxBackColor	ImgTxBoxLines
ImgTxCaption	ImgTxLedBackColor	ImgTxLedColor
ImgTxShowDetails	ImgTxShowLed	ObjectCount
ObjectLocalCount	ObjectNumber	PIDsInfo
PointsInfo	ScanActive	ScanAutoTrigger

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

ScanInterval	ScanPriority	ScanRetries
ScanStart	ScanStopAfter	TcplpEnabled
TcplpLocalPort	TcplpProtocol	TcplpRemoteAddress
TcplpRemotePort	TcplpTimeout	TimerInterval

## Métodos

AbortCommunication()	CommOwner(commport)
CommTotalCount(commport)	CommUseCount(commport)
FindPoint(pointname)	ForcePIDCalculation(point)
GetCalculatedDate(BaseDate, Seconds)	GetCurrentDate()
GetCurrentRetry()	GetDateDifference(LastDate, FirstDate)
GetDateMilliseconds(Date)	GetDriverValue(point)
GetHAlarmColor(point)	GetHHAAlarmColor(point)
GetHRAAlarmColor(point)	GetLAlarmColor(point)
GetLLAlarmColor(point)	GetLRAAlarmColor(point)
InitPointValue(point, value)	PointCacheDefaultValue(point)
PointCacheDirection(point)	PointCacheEnabled(point)
PointCachePeriod(point)	PointCachePointer(point)
PointCacheSamples(point)	PointCacheSampleValue(point, sample)
PointCacheSource(point)	PointCalculationsEnabled(point)
PointDeadBand(point)	PointDeltaTime(point)
PointDeltaValue(point)	PointHAlarmBlinkEnabled(point)
PointHAlarmCount(point)	PointHAlarmDate(point)
PointHAlarmDeclared(point)	PointHAlarmElapsedTime(point)
PointHAlarmEnabled(point)	PointHAlarmHysteresis(point)
PointHAlarmMessage(point)	PointHAlarmNotified(point)
PointHAlarmNotifiedCount(point)	PointHAlarmNotifiedDate(point)
PointHAlarmThreshold(point)	PointHAlarmTimeTolerance(point)
PointHAlarmToleranceExceeded(point)	PointHHAAlarmBlinkEnabled(point)
PointHHAAlarmCount(point)	PointHHAAlarmDate(point)
PointHHAAlarmDeclared(point)	PointHHAAlarmElapsedTime(point)
PointHHAAlarmEnabled(point)	PointHHAAlarmHysteresis(point)
PointHHAAlarmMessage(point)	PointHHAAlarmNotified(point)
PointHHAAlarmNotifiedCount(point)	PointHHAAlarmNotifiedDate(point)
PointHHAAlarmThreshold(point)	PointHHAAlarmTimeTolerance(point)
PointHHAAlarmToleranceExceeded(point)	PointHighDriverValue(point)
PointHighHardLimit(point)	PointHighHardLimitDate(point)
PointHighPhysicValue(point)	PointHRAAlarmBlinkEnabled(point)
PointHRAAlarmCount(point)	PointHRAAlarmDate(point)
PointHRAAlarmDeclared(point)	PointHRAAlarmElapsedTime(point)
PointHRAAlarmEnabled(point)	PointHRAAlarmHysteresis(point)
PointHRAAlarmMessage(point)	PointHRAAlarmNotified(point)
PointHRAAlarmNotifiedCount(point)	PointHRAAlarmNotifiedDate(point)
PointHRAAlarmThreshold(point)	PointHRAAlarmTimeTolerance(point)
PointHRAAlarmToleranceExceeded(point)	PointIntegral(point)
PointLAlarmBlinkEnabled(point)	PointLAlarmCount(point)
PointLAlarmDate(point)	PointLAlarmDeclared(point)
PointLAlarmElapsedTime(point)	PointLAlarmEnabled(point)
PointLAlarmHysteresis(point)	PointLAlarmMessage(point)
PointLAlarmNotified(point)	PointLAlarmNotifiedCount(point)
PointLAlarmNotifiedDate(point)	PointLAlarmThreshold(point)
PointLAlarmTimeTolerance(point)	PointLAlarmToleranceExceeded(point)
PointLastScanDate(point)	PointLLAlarmBlinkEnabled(point)
PointLLAlarmCount(point)	PointLLAlarmDate(point)
PointLLAlarmDeclared(point)	PointLLAlarmElapsedTime(point)
PointLLAlarmEnabled(point)	PointLLAlarmHysteresis(point)

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

PointLLAlarmMessage(point)	PointLLAlarmNotified(point)
PointLLAlarmNotifiedCount(point)	PointLLAlarmNotifiedDate(point)
PointLLAlarmThreshold(point)	PointLLAlarmTimeTolerance(point)
PointLLAlarmToleranceExceeded(point)	PointLowDriverValue(point)
PointLowHardLimit(point)	PointLowHardLimitDate(point)
PointLowPhysicValue(point)	PointLRAlarmBlinkEnabled(point)
PointLRAlarmCount(point)	PointLRAlarmDate(point)
PointLRAlarmDeclared(point)	PointLRAlarmElapsedTime(point)
PointLRAlarmEnabled(point)	PointLRAlarmHysteresis(point)
PointLRAlarmMessage(point)	PointLRAlarmNotified(point)
PointLRAlarmNotifiedCount(point)	PointLRAlarmNotifiedDate(point)
PointLRAlarmThreshold(point)	PointLRAlarmTimeTolerance(point)
PointLRAlarmToleranceExceeded(point)	PointName(point)
PointNextScanDate(point)	PointPIDControlSignal(point)
PointPIDControlSignalDate(point)	PointPIDDeadBand(point)
PointPIDEnabled(point)	PointPIDError(point)
PointPIDErrorIntegral(point)	PointPIDErrorRateOfChange(point)
PointPIDHighPhysic(point)	PointPIDKd(point)
PointPIDKi(point)	PointPIDKp(point)
PointPIDLowPhysic(point)	PointPIDMaxControlSignal(point)
PointPIDMinControlSignal(point)	PointPIDMode(point)
PointPIDSetPoint(point)	PointPIDSetPointAsPercentage(point)
PointPreviousRateOfChange(point)	PointPreviousValue(point)
PointPreviousValueDate(point)	PointPWMPeriod(point)
PointPWMSetpoint(point)	PointRateOfChange(point)
PointRateOfChangeDeadBand(point)	PointScanTime(point)
PointUnits(point)	PointValue(point)
PointValueDate(point)	ReadComm(header, footer, size, timeout)
ResetCache(point)	ResetPending()
ResetPWM(point)	ScaleDriverValue(point, value)
ScrollCache(point, value)	SetCommBreak(Status)
SetCommDTR(Status)	SetCommRTS(Status)
SetInitialPointValue(point, value)	Setm_timeOffset(offset)
ShowAboutPropertyPage()	ShowAlarmsPropertyPage()
ShowCachesPropertyPage()	ShowColorsPropertyPage()
ShowCommPropertyPage()	ShowCPUPropertyPage()
ShowDriverPropertyPage()	ShowFontsPropertyPage()
ShowImagePropertyPage()	ShowInfoPropertyPage()
ShowPIDsPropertyPage()	ShowPointsPropertyPage()
ShowPropertyPages()	ShowScanPropertyPage()
ShowTcplpPropertyPage()	TransmitBuffer(buffer, length)
TransmitMessage(message)	Trigger()
TriggerAndWait()	TriggerWithDeadBand()
Wait(timeout)	WaitForMessage(message, timeout)
WakeUp()	

## Eventos

Click()	DbiClick()
OnAfterRequest()	OnAnyPointValueChanged()
OnBeforeRequest()	OnCommRxBufferChanged()
OnCommTxBufferChanged()	OnDriverEfficiencyChanged()
OnDriverEfficiencyDecreased()	OnDriverEfficiencyIncreased()
OnDriverStatusChanged()	OnErrorCleared()
OnErrorDeclared()	OnErrorReceiving()
OnErrorSending()	OnHAlarmBlinking(point)
OnHAlarmCleared(point)	OnHAlarmDeclared(point)

CPKSoft Ingeniería  
Drivers para comunicación industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

OnHAlarmElapsedTimeChanged(point)	OnHAlarmNotified(point)
OnHAlarmToleranceExceeded(point)	OnHighHardLimitReached(point)
OnHLAlarmBlinking(point)	OnHLAlarmCleared(point)
OnHLAlarmDeclared(point)	OnHLAlarmElapsedTimeChanged(point)
OnHLAlarmNotified(point)	OnHLAlarmToleranceExceeded(point)
OnHRAAlarmBlinking(point)	OnHRAAlarmCleared(point)
OnHRAAlarmDeclared(point)	OnHRAAlarmElapsedTimeChanged(point)
OnHRAAlarmNotified(point)	OnHRAAlarmToleranceExceeded(point)
OnLAlarmBlinking(point)	OnLAlarmCleared(point)
OnLAlarmDeclared(point)	OnLAlarmElapsedTimeChanged(point)
OnLAlarmNotified(point)	OnLAlarmToleranceExceeded(point)
OnLLAlarmBlinking(point)	OnLLAlarmCleared(point)
OnLLAlarmDeclared(point)	OnLLAlarmElapsedTimeChanged(point)
OnLLAlarmNotified(point)	OnLLAlarmToleranceExceeded(point)
OnLowHardLimitReached(point)	OnLRAAlarmBlinking(point)
OnLRAAlarmCleared(point)	OnLRAAlarmDeclared(point)
OnLRAAlarmElapsedTimeChanged(point)	OnLRAAlarmNotified(point)
OnLRAAlarmToleranceExceeded(point)	OnPointCacheChanged(point)
OnPointControlSignalChanged(point)	OnPointControlSignalDecreased(point)
OnPointControlSignalIncreased(point)	OnPointIntegralChanged(point)
OnPointPWMOutputChanged(point)	OnPointRateOfChangeChanged(point)
OnPointRateOfChangeDecreased(point)	OnPointRateOfChangeIncreased(point)
OnPointReset(point)	OnPointSet(point)
OnPointValueChanged(point)	OnPointValueDecreased(point)
OnPointValueIncreased(point)	OnRequestCompleted(out retry)
OnRequestError()	OnRequestRetried()
OnRequestStart(boolean* Cancel)	OnRxLedOff()
OnRxLedOn()	OnScanStopped()
OnSuccessfullyReceived()	OnSuccessfullySent()
OnTimer()	OnTxLedOff()
OnTxLedOn()	

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/](https://www.facebook.com/cpksoftingenieria)

cpksoftingenieria

cpksoftingenieria@

hotmail.com

tel: 54-11-1545788354

1990-2012

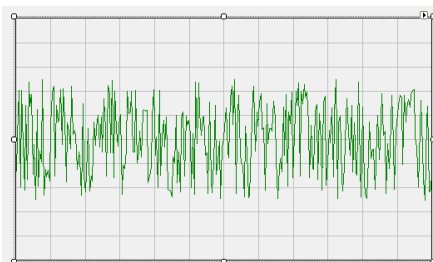


# Soluciones de comunicación industrial para Windows

## Uso del componente LineChart

### La interfaz de LineChart

El componente LineChart es un control ActiveX/OCX con interfaz tipo OLE/COM que se distribuye con el driver para dar compatibilidad a aplicaciones basadas en HMITalk que lo han venido utilizando y que necesiten ser migradas a las nuevas plataformas soportadas. Su finalidad es permitir armar gráficos sencillos de hasta cuatro plumas para visualizar la evolución de los datos que se van recibiendo desde el driver, ya sea que se hayan obtenido mediante el componente HMITalk o NetTalk. En realidad, el componente se puede utilizar para graficar datos provenientes de cualquier fuente, si bien fue originalmente diseñado para trabajar en conjunto con HMITalk.

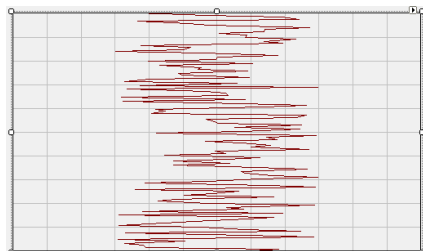


LineChart permite colocar una grilla de tamaño, densidad y color configurable detrás de las plumas graficadas.

La cantidad de muestras permitidas es hasta 2400, siendo 120 por defecto. Las muestras se van empujando dentro del gráfico, lo que produce un efecto de avance o scroll.

Se pueden definir zonas de diferente color que sirvan para indicar visualmente que las plumas están dentro de rangos no permitidos.

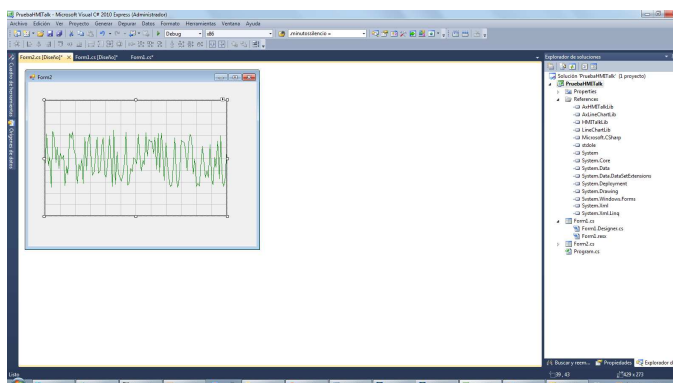
Las plumas pueden ser horizontales o verticales, y moverse de derecha a izquierda o de abajo hacia arriba.



Por tratarse de un control Activex/OCX, debe también ser registrado antes de ser utilizado. El programa RegHMITalk.exe, utilizado para registrar HMITalk, también se encarga de registrar LineChart tanto para 32 bits como para 64 bits y son válidas todas las explicaciones ya dadas sobre este punto al explicar el componente HMITalk.

### Dibujar una pluma

Para dibujar una pluma que muestre datos, primero debemos colocar una instancia del objeto en cualquier formulario:



CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

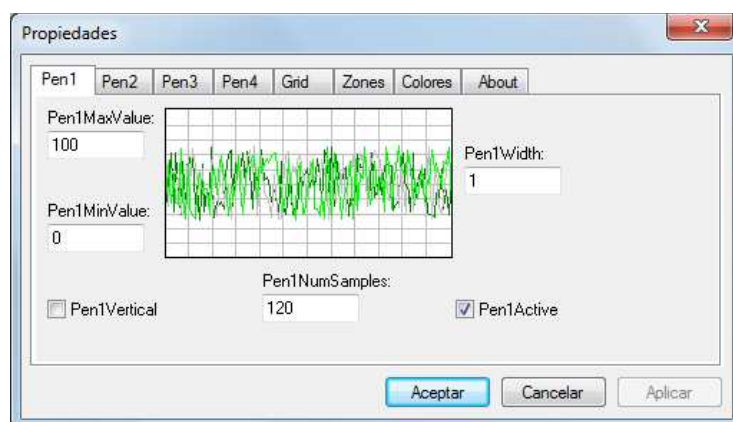
www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012



# Soluciones de comunicación industrial para Windows

Con botón derecho sobre el objeto, se debe acceder a la pestaña Pen1 de las páginas de propiedades, donde se le indica al objeto la cantidad de muestras o puntos que contendrá la pluma, los rangos máximo y mínimo, el espesor de la pluma y si la misma es horizontal o vertical:

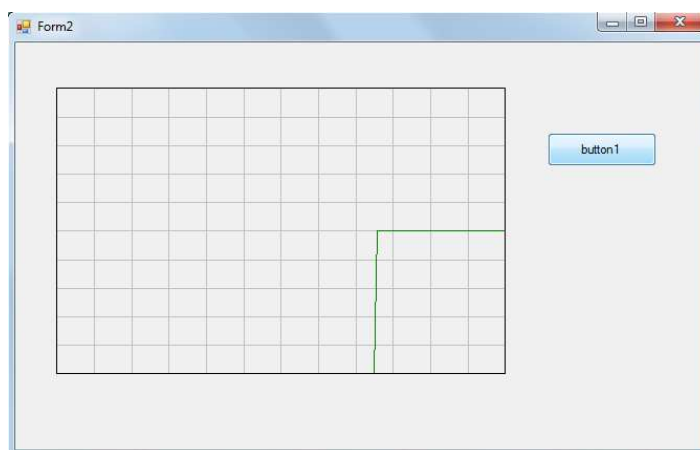


Adicionalmente se puede ir a la pestaña Colores y modificar el color de la pluma. Si se quieren colocar más plumas, se pueden colocar hasta cuatro, activándolas en las pestañas correspondientes.

El código recomendado para ir cargando valores a ser mostrados en la pluma Pen1 es el siguiente, por ejemplo utilizando un botón para cargar un valor 50:

```
private void button1_Click(object sender, EventArgs e)
{
    this.axLineChart1.ScrollPen1(50);
    this.axLineChart1.Draw();
}
```

El resultado luego de presionar repetidas veces el botón será el siguiente:



Es decir, cada vez que se llame al método ScrollPen1, la pluma 1 empujará todas las muestras existentes de derecha a izquierda y hará lugar para una nueva muestra con el valor indicado en su extremo derecho. El resultado será un gráfico que irá avanzando mostrando la historia reciente.

El método .Draw() es quien produce la actualización del gráfico. Se lo llama al final, cuando las muestras ya han sido cargadas en el buffer del objeto. La razón de separar la carga de los valores del refresco del gráfico es para permitir actualizar varias muestras o varias plumas, y que mientras tanto no se produzca un efecto de flickering o parpadeo del gráfico.

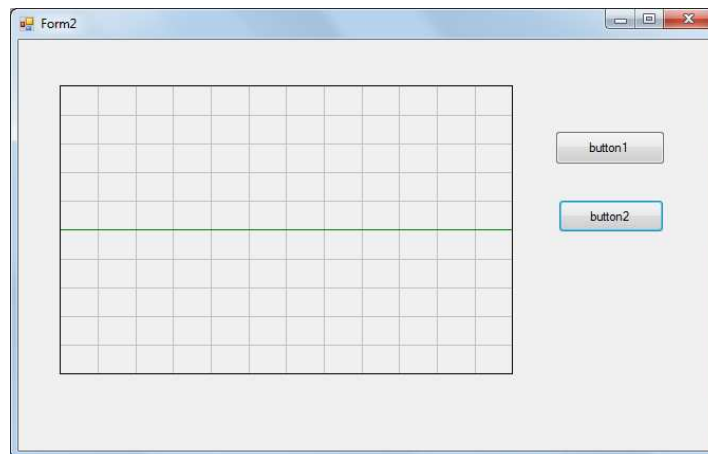
# Soluciones de comunicación industrial para Windows

Una alternativa para llenar una pluma es mediante el método `SetPenValue`, que permite establecer el valor de cualquier muestra del gráfico en cualquier momento. Es ideal para transferir de una sola vez un conjunto completo de muestras al gráfico, en lugar de hacer scrollear el gráfico con cada muestra.

Por ejemplo, este código carga de punta a punta toda la pluma con un valor 50, llamando al método `Draw()` al terminar la carga:

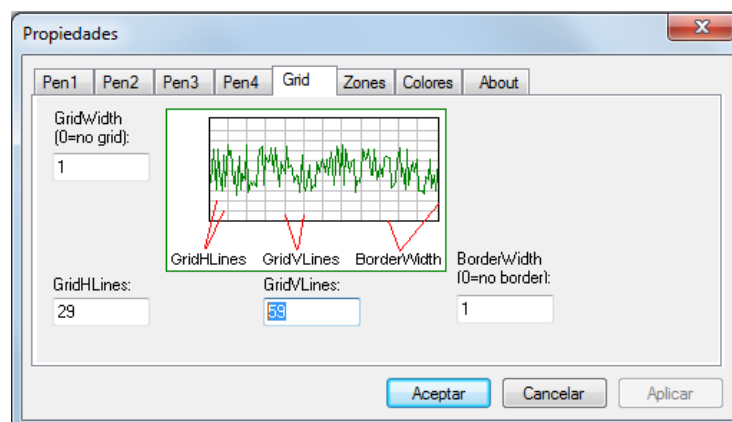
```
private void button2_Click(object sender, EventArgs e)
{
    for (short i = 0; i < this.axLineChart1.Pen1NumSamples; i++)
    {
        this.axLineChart1.SetPenValue(0, i, 50);
    }
    this.axLineChart1.Draw();
}
```

El resultado que se obtiene es el siguiente:



## Manejo de la grilla de fondo

La grilla de fondo se puede ajustar para que tenga mas o menos divisiones, tanto horizontal como verticalmente. Para ello se utiliza la pestaña `Grid` de las páginas de propiedades del objeto:



CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

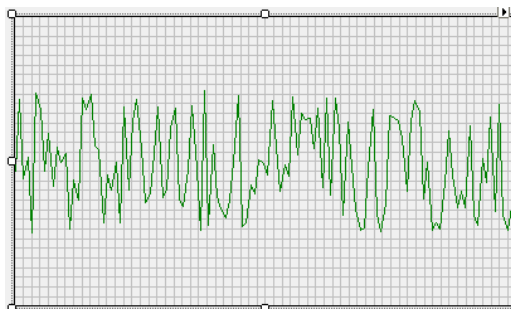
www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

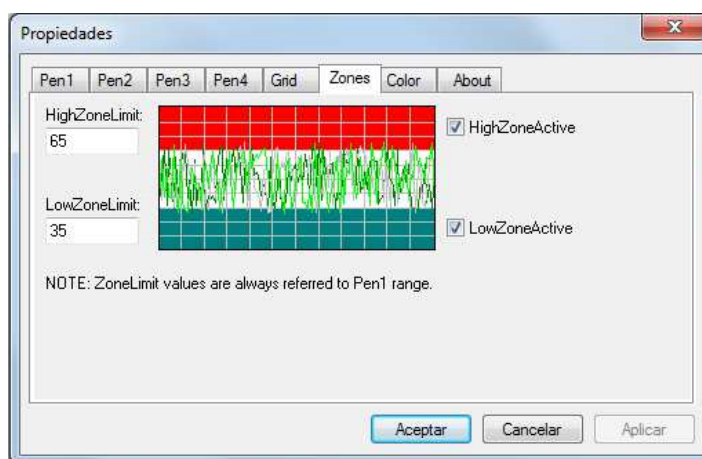
La propiedad GridHLines define cuantas líneas horizontales tendrá la grilla. Por su parte, GridVLines define la cantidad de líneas verticales. Obsérvese que se trata de cantidad de líneas, y no de cantidad de divisiones o de espacios entre líneas. Por este motivo, si se quisiera tener 60 divisiones para representar los últimos 60 minutos o 60 segundos, se debe indicar un valor 59 en GridVLines.

Por ejemplo, aquí se definieron 29 líneas horizontales y 59 líneas verticales. Con esto se obtienen 30 divisiones a lo alto y 60 divisiones a lo ancho:

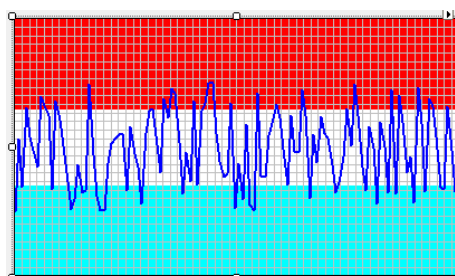


## Bandas de color

LineChart permite definir una zona o banda superior y otra inferior de diferentes colores que sirven para establecer visualmente si determinadas porciones del gráfico de una o varias plumas se encuentran fuera de un rango. Para establecer estas zonas, se utiliza la pestaña Zonas de la página de propiedades:



Cada zona se puede activar individualmente y su tamaño se define en relación al rango establecido para la pluma 1 (Pen1). Sus colores se establecen en la pestaña Color. En la imagen se estableció una zona verde de 0 a 35 y una zona roja de 65 a 100, teniendo en cuenta que la pluma 1 estaba definida con un rango de 0 a 100.



CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/cpksoftingenieria](http://www.facebook.com/cpksoftingenieria)

[cpksoftingenieria@hotmail.com](mailto:cpksoftingenieria@hotmail.com)

tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

## Guía rápida de propiedades, métodos y eventos

### Propiedades

BackColor	BorderColor	BorderWidth
GridColor	GridHLines	GridVLines
GridWidth	HighZoneActive	HighZoneColor
HighZoneLimit	LowZoneActive	LowZoneColor
LowZoneLimit	Pen1Active	Pen1Color
Pen1MaxValue	Pen1MinValue	Pen1NumSamples
Pen1Vertical	Pen1Width	Pen2Active
Pen2Color	Pen2MaxValue	Pen2MinValue
Pen2NumSamples	Pen2Vertical	Pen2Width
Pen3Active	Pen3Color	Pen3MaxValue
Pen3MinValue	Pen3NumSamples	Pen3Vertical
Pen3Width	Pen4Active	Pen4Color
Pen4MaxValue	Pen4MinValue	Pen4NumSamples
Pen4Vertical	Pen4Width	

### Métodos

Draw()	FillPen(pen, hmitalkcachepointer)
FillPen1(hmitalkcachepointer)	FillPen2(hmitalkcachepointer)
FillPen3(hmitalkcachepointer)	FillPen4(hmitalkcachepointer)
GetPenAverage(pen)	GetPenValue(pen, sample)
ScrollPen(pen, value)	ScrollPen1(value)
ScrollPen2(value)	ScrollPen3(value)
ScrollPen4(value)	SetPenValue(pen, sample, value)
ShowAboutPropertyPage()	ShowColorsPropertyPage()
ShowGridPropertyPage()	ShowPen1PropertyPage()
ShowPen2PropertyPage()	ShowPen3PropertyPage()
ShowPen4PropertyPage()	ShowPropertyPages()
ShowZonesPropertyPage()	

### Eventos

Click
DbtClick
MouseDown
MouseMove
MouseUp

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/cpksoftingenieria](https://www.facebook.com/cpksoftingenieria)

[cpksoftingenieria@hotmail.com](mailto:cpksoftingenieria@hotmail.com)

[cpksoftingenieria@hotmail.com](mailto:cpksoftingenieria@hotmail.com)

tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

## Distribución de las aplicaciones

### Archivos a ser distribuidos

Para distribuir una aplicación escrita con NetTalk se deben copiar en la máquina de destino todos los archivos requeridos para que la misma se ejecute correctamente. Estos archivos son básicamente el .exe de la aplicación, el archivo NetTalk.dll del componente NetTalk, el archivo .tlk del driver, y el archivo de licencia runtime .lic del driver. Este último es generado por el propietario de la licencia ilimitada en función de la identificación de la máquina de destino.

Adicionalmente, se deberá constatar que la máquina de destino tenga previamente instalada la versión del Framework.NET requerida por la aplicación. La distribución del framework no es responsabilidad de la aplicación, aunque ésta podría venir acompañada de todos los ensamblados que se utilicen para asegurarse que no falten. No obstante, es común que Windows ya tenga algún framework preinstalado. Por ejemplo, Windows 7 ya viene con el Framework 3.5. El componente NetTalk sólo necesita la versión 2.0 del framework para correr sin problemas.

Los archivos que deben ser distribuidos cuando se construye una aplicación utilizando el componente NetTalk son:

- **[Su aplicación].exe:** El ejecutable de la aplicación.
- **NetTalk.dll:** es el archivo del ensamblado que corresponde al componente NetTalk, quien actúa como interfaz para manejar el driver XFINS.tlk desde aplicaciones .NET.
- **XFINS.32.tlk:** archivo del driver para plataforma de 32 bits.
- **XFINS.64.tlk:** archivo del driver para plataforma de 64 bits.
- **XFINS.<PCId>.lic:** es el archivo con la licencia runtime del driver para la máquina de destino cuyo número de identificación es "<PCId>". Esta licencia es válida tanto para 32 bits como para 64 bits.
- **RegNetTalk.exe:** utilitario opcional para registrar la interfaz COM de NetTalk, en el caso en que se utilizó como objeto tipo COM desde la aplicación. Se puede reemplazar por un programa de instalación propio que realice el registro del ensamblado o bien con una explicación al cliente para que lo registre él mismo con regasm.exe.

Los archivos que deben ser distribuidos cuando se construye una aplicación utilizando el componente HMITalk son:

- **[Su aplicación].exe:** El ejecutable de la aplicación.
- **HMITK32.ocx y HMITK64.ocx:** los archivos del ActiveX HMITalk, para 32 y 64 bits. Se recomienda distribuir ambas versiones, especialmente si la aplicación se compiló con la opción "Any CPU".
- **XFINS.32.tlk:** archivo del driver para plataforma de 32 bits.
- **XFINS.64.tlk:** archivo del driver para plataforma de 64 bits.
- **XFINS.<PCId>.lic:** es el archivo con la licencia runtime del driver para la máquina de destino cuyo número de identificación es "<PCId>". Esta licencia es válida tanto para 32 bits como para 64 bits.
- **GRID32.ocx:** sólo si la aplicación de destino es 32 bits y se invocan en runtime alguno de los métodos que muestran las páginas de propiedades del objeto HMITalk.
- **LCHART32.ocx y LCHART64.ocx:** sólo si la aplicación utiliza el objeto LineChart. Se recomienda distribuir ambas versiones.
- **RegHMITalk.exe:** utilitario opcional para registrar HMITalk. Se puede reemplazar por un programa de instalación propio que realice el registro o bien con una explicación al cliente para que lo registre él mismo con el utilitario regsvr32.exe que corresponda a la plataforma de destino.
- **AxInterop.HMITalkLib.dll / Interop.HMITalkLib.dll / AxInterop.LineChartLib.dll / Interop.LineChartLib.dll:** requeridos por los objetos HMITalk y LineChart para su funcionamiento cuando se crean aplicaciones en .NET. Se sugiere utilizar los que genera automáticamente el compilador en la carpeta bin\Release de la aplicación. Se debe tener en cuenta que cuando se compila para 32 bits, 64 bits o AnyCPU, estos archivos mantienen los mismos nombres pero sus contenidos son diferentes, por lo que se debe tener la precaución de distribuir las versiones correctas según la CPU de destino.

# Soluciones de comunicación industrial para Windows

## Generación de licencias runtime

### Explicación del proceso

Cada vez que se deba correr una aplicación escrita con NetTalk, cada driver utilizado que no sea gratuito chequeará la existencia y validez de un archivo de licencia runtime de extensión .lic que debe ser creado previamente por el poseedor de la licencia ilimitada del driver.

Los archivos de licencias runtime .lic son válidos para un driver en particular y para una máquina en particular. El poseedor de la licencia ilimitada puede crear tantos archivos .lic como desee y para cada driver que se utilice en las aplicaciones se deberá crear un .lic individual para cada máquina de destino donde deban correr esas aplicaciones.

Para poder generar licencias runtime, se necesitan los siguientes elementos:

- Un número de licencia ilimitada, otorgado al adquirir el producto.
- Un número de identificación de la máquina de destino, recolectado con GetPCId.exe.
- Un utilitario para generar los archivos .lic, MakeLic.exe entregado con el producto.

### Obtención del número de licencia ilimitada

Cuando usted adquiere una licencia ilimitada de cualquiera de nuestros drivers no gratuitos, entre los archivos entregados usted recibe un documento .pdf protegido con password conteniendo los datos de su licencia.

Utilizando como clave el correo electrónico que nos suministró en su compra podrá abrir el documento, que se verá como sigue:



CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

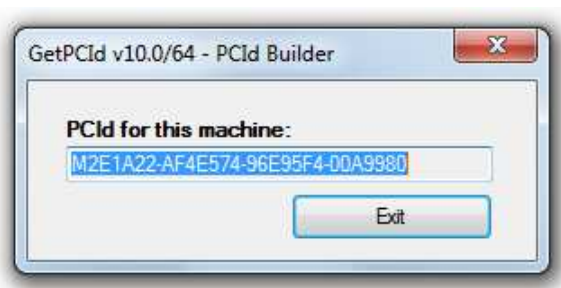
1990-2012

# Soluciones de comunicación industrial para Windows

En el centro del documento, encontrará su número de licencia ilimitada, que es único para el driver adquirido y para su empresa. Este número debe ser bien custodiado por usted y jamás revelado a sus clientes. Se utiliza para activar la herramienta de armado de licencias runtime.

## Obtención del PCId mediante el utilitario GetPCId.exe

Para obtener el PCId de una máquina con el utilitario GetPCId.exe sólo se necesita correr este pequeño ejecutable que no requiere de ninguna instalación previa en la máquina de su cliente. Al correrlo, el programa le mostrará una ventana donde le aparecerá un número como se ve en la imagen siguiente:



El utilitario puede ser corrido por usted si es que tiene acceso a la computadora de destino, o bien puede enviarlo a su cliente para que sea él quien lo corra. En cualquier caso, usted deberá hacerse del número obtenido ya que luego le será necesario para generar la licencia runtime correspondiente a esa máquina. El utilitario deliberadamente no muestra ninguna referencia a CPKSoft ni a NetTalk ó HMITalk.

El número obtenido es válido para correr aplicaciones tanto de 32 bits como de 64 bits.

En la ventana mostrada por GetPCId, el número completo ya aparece pre-seleccionado para que sea fácil copiarlo y pegarlo en un correo o directamente en el utilitario MakeLic.exe de generación de licencias.

## Obtener PCId por código desde la aplicación

Una forma alternativa de obtener el PCId de una máquina es llamar al método **GetMachineKey()** de NetTalk o bien consultar la propiedad **PCId** desde componente NetTalk. La aplicación debe estar corriéndose en la máquina cuyo PCId se desee obtener.

De esta forma, la obtención del PCId puede hacerse sin la intervención directa del cliente. Usted puede hacer que la aplicación chequee la existencia del archivo de licencia XFINS.<PCId>.lic, y en caso de no encontrarse, le despliegue un mensaje al cliente mostrando el PCId con instrucciones para hacérselo llegar a usted.

## Creación del archivo .lic

Para generar las licencias runtime a partir de su número de licencia ilimitada y del número de identificación de la máquina de destino, se utiliza el programa MakeLic.exe que se distribuye únicamente con las versiones licenciadas de cada driver. Este programa no está presente en las versiones de evaluación y prueba de los drivers.

Los archivos generados de extensión .lic y su nombre está formado por el nombre del driver licenciado y el PCId de la máquina de destino, como en el ejemplo siguiente:

XMODBTCP.M73711E-3ACA5FD-1268560-00A3A10.lic.

Ese archivo .lic es válido para utilizar sin restricciones al driver XMODBTCP en una máquina cuyo PCId sea M73711E-3ACA5FD-1268560-00A3A10. El contenido del archivo consiste en información interna sobre el driver, datos del creador del mismo, fecha de expiración (si la tiene) y un código de validación.

La interfaz del utilitario MakeLic, que no se debe distribuir a los usuarios, es la siguiente:

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354



# Soluciones de comunicación industrial para Windows

CPKSoft Engineering MakeLic v10.0/64 - Driver Runtime Licenses Generator

**Unlimited License Owner Information**

Company name: CPKSoft Engineering

Customer name: Carlos Perez Kuper

Customer e-mail address: cperezkuper@hotmail.com

Order number: 12345

**Driver to license**

Driver name: XFINS.TLK

Description: Omron SYSMAC Host Link FINS Protocol Driver

Version: 9.0

Developer: CPKSoft Engineering

Enter your Unlimited License Number for this driver (available in the .pdf):

Expiration date (optional): MM None / DD None / YYYY None

**Destination machine**

Enter the PCId machine number collected with WGetPCId.exe:

M19CEA0-01BAFA0-01BEE20-007E92C

GetPCId

Generate Runtime License

Exit

Este utilitario se debe utilizar para cada driver y para cada máquina de destino en la que se deba correr la aplicación generada con NetTalk o HMITalk. Los archivos .lic resultantes deben acompañar a la aplicación a la máquina de destino correspondiente, y se deben copiar en la misma carpeta donde residan los .tlk de cada driver.

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

## Especificaciones técnicas del driver XFINS

### Información general

XFINS driver allows you to connect to the OMRON SYSMAC CS/CJ/CP Series and SYSMAC One NSJ Series programmable controllers which use the FINS commands via the Host Link serial communication protocol.

This driver does not support slave-initiated communications. All communications are always initiated by the host PC.

### Listado de comandos

#### Memory Area Read

**Descripción del comando:**

Reads the contents of the specified number of consecutive memory area items starting from the specified address.

**Métodos usados para ejecutar este comando:**

Analog Input / Digital Input

**Número de puntos permitidos para este comando:**

1-250

**Significado del parámetro P0:**

Identifies the CPU Unit connected to the host PC. (See note below).

**Significado del parámetro P1:**

Response wait time, in milliseconds (0-150). (See note below).

**Significado del parámetro P2:**

Identifies the Destination Network Address of the target PLC (0-127). Use -1 if the target is the directly connected PLC.

**Significado del parámetro P3:**

Identifies the Destination Node Address of the target PLC (0-254). Leave empty or 0 if the target is the directly connected PLC. (See note below).

**Significado del parámetro P4:**

Identifies the Memory Area Code to be read. (See tables for CS/CJ/CV below).

**Significado del parámetro P5:**

Identifies the starting Memory Address to be read. (See tables for CS/CJ/CV below).

**Valores que son devueltos:**

Valor del punto (0) = First item value read (0-65535)

Valor del punto (1) = Second item value read (0-65535)

...

Valor del punto (n-1) = Last item value read (0-65535)

**NOTE ABOUT DIGITAL INPUTS:**

When DriverDataType is set to Digital Input, values returned are forced to 0 if value received was  $\leq 0$ , or to 1 if value received was  $> 0$ , regardless of the memory Area Code or Item Data Type being read.

**NOTE ABOUT UNIT NUMBER:**

The unit number set in DriverP0 is that of the destination CPU Unit connected to the host computer. When the host computer is connected to a CPU Unit, the unit number is designated in the PLC Setup. When the host computer is connected to a Serial Communications Board or a Serial Communications Unit, the unit number is the designated in the Setup for the Board or Unit. DriverP0 accepts values from 0 to 99.

**NOTE ABOUT RESPONSE WAIT TIME:**

The response wait time set in DriverP1 sets the time from when the CPU Unit receives a command block until it starts to return a response. It can be set from 0 to 150 ms, in units of 10 ms.

**NOTE ABOUT DESTINATION NODE ADDRESS:**

The destination node address is set in DriverP3 and it is required only when sending to a CPU Unit on a network. Set it to -1 if the PLC to be read is the same that is directly connected to the host PC.

*Must be set within the following ranges:*

- PLC directly connected to the Host PC = -1

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/cpksoftingenieria](https://www.facebook.com/cpksoftingenieria)

[cpksoftingenieria@](mailto:cpksoftingenieria@hotmail.com)

hotmail.com

tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

- Ethernet Units with model numbers ending in ETN21 = 1 to 254
- Ethernet Units with other model numbers = 1 to 126
- Controller Link Unit = 1 to 32
- SYSMAC NET = 1 to 126
- SYSMAC LINK = 1 to 62

## MEMORY ADDRESS TABLE FOR CS/CJ MODE:

Select the data type to be read and set the Memory Area Code in DriverP4 and the Memory Address in DriverP5.

- CIO Bit: Memory Area Code = 30h, Memory Address for CIO 000000 to CIO 614315 = 000000h to 17FF0Fh
- WR Bit: Memory Area Code = 31h, Memory Address for W00000 to W51115 = 000000h to 01FF0Fh
- HR Bit: Memory Area Code = 32h, Memory Address for H00000 to H51115 = 000000h to 01FF0Fh
- AR Bit (read only): Memory Area Code = 33h, Memory Address for A00000 to A44715 = 000000h to 01BF0Fh
- AR Bit (read/write): Memory Area Code = 33h, Memory Address for A44800 to A95915 = 01C000h to 03BF0Fh
- CIO Word: Memory Area Code = B0h, Memory Address for CIO 0000 to CIO 6143 = 000000h to 17FF00h
- WR Word: Memory Area Code = B1h, Memory Address for W000 to W511 = 000000h to 01FF00h
- HR Word: Memory Area Code = B2h, Memory Address for H000 to H511 = 000000h to 01FF00h
- AR Word (read only): Memory Area Code = B3h, Memory Address for A000 to A447 = 000000h to 01BF00h
- AR Word (read /write): Memory Area Code = B3h, Memory Address for A448 to A959 = 01C000h to 03BF00h
- TIM Completion Flag: Memory Area Code = 09h, Memory Address for T0000 to T4095 = 000000h to 0FFF00h
- CNT Completion Flag: Memory Area Code = 09h, Memory Address for C0000 to C4095 = 800000h to 8FFF00h
- TIM PV: Memory Area Code = 89h, Memory Address for T0000 to T4095 = 000000h to 0FFF00h
- CNT PV: Memory Area Code = 89h, Memory Address for C0000 to C4095 = 800000h to 8FFF00h
- DM Bit: Memory Area Code = 02h, Memory Address for D0000000 to D3276715 = 000000h to 7FFF0Fh
- DM Word: Memory Area Code = 82h, Memory Address for D00000 to D32767 = 000000h to 7FFF00h
- EM Bank 0 to Bank F Bit: Memory Area Code = 20h to 2Fh, Memory Address for E0\_0000000 to 3276715 / EF\_0000000 to 3276715 = 000000h to 7FFF0Fh
- EM Bank 10 to Bank 18: Memory Area Code = E0h to E8h, Memory Address for E10\_0000000 to 3276715 / E18\_0000000 0 to 3276715 = 000000h to 7FFF0Fh
- EM Bank 0 to Bank F Word: Memory Area Code = A0h to AFh or 50h to 5Fh, Memory Address for E0\_00000 to 32767 / EF\_00000 to 32767 = 000000h to 7FFF00h
- EM Bank 10 to Bank 18 Word: Memory Area Code = 60h to 68h, Memory Address for E10\_00000 to 32767 / E18\_00000 to 32767 = 000000h to 7FFF00h
- EM Current Bank Bit: Memory Area Code = 0Ah, Memory Address for E0000000 to E3276715 = 000000h to 7FFF0Fh
- EM Current Bank Word: Memory Area Code = 98h, Memory Address for E00000 to E32767 = 000000h to 7FFF00h
- EM Current Bank No. Word: Memory Area Code = BCh, Memory Address = 0F0000h
- TK Bit: Memory Area Code = 06h, Memory Address for TK0000 to TK0031 = 000000h to 001F00h
- TK Status Bit: Memory Area Code = 46h, Memory Address for TK0000 to TK0031 = 000000h to 001F00h
- IR PV: Memory Area Code = DCh, Memory Address for IR00 to IR15 = 010000h to 010F00h
- DR PV: Memory Area Code = BCh, Memory Address for DR00 to DR15 = 020000h to 020F00h

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/cpksoftingenieria](https://www.facebook.com/cpksoftingenieria)

[cpksoftingenieria@](mailto:cpksoftingenieria@hotmail.com)

[hotmail.com](mailto:hotmail.com)

tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

## MEMORY ADDRESS TABLE FOR CV MODE:

Select the data type to be read and set the Memory Area Code in DriverP4 and the Memory Address in DriverP5.

- CIO Bit: Memory Area Code = 00h, Memory Address for CIO 000000 to CIO 255515 = 000000h to 09FB0Fh
- AR Bit (read only): Memory Area Code = 00h, Memory Address for A00000 to A44715 = 0B0000h to 0CBF0Fh
- AR Bit (read/write): Memory Area Code = 00h, Memory Address for A44800 to A95915 = 0CC000h to 0EBF0Fh
- CIO Word: Memory Area Code = 80h, Memory Address for CIO 0000 to CIO 2555 = 000000h to 09FB00h
- AR Word (read only): Memory Area Code = 80h, Memory Address for A000 to A447 = 0B0000h to 0CBF00h
- AR Word (read /write): Memory Area Code = 80h, Memory Address for A448 to A959 = 0CC000h to 0EBF00h
- TIM Completion Flag: Memory Area Code = 01h, Memory Address for T0000 to T2047 = 000000h to 07FF00h
- CNT Completion Flag: Memory Area Code = 01h, Memory Address for C0000 to C2047 = 080000h to 0FFF00h
- TIM PV: Memory Area Code = 81h, Memory Address for T0000 to T2047 = 000000h to 07FF00h
- CNT PV: Memory Area Code = 81h, Memory Address for C0000 to C2047 = 080000h to 0FFF00h
- DM Word: Memory Area Code = 82h, Memory Address for D00000 to D32767 = 000000h to 7FFF00h
- EM Bank 0 to Bank F Word: Memory Area Code = 90h to 97h, Memory Address for E0\_00000 to 32767 / E7\_00000 to 32767 = 000000h to 7FFF00h
- EM Current Bank Word: Memory Area Code = 98h, Memory Address for E00000 to E32767 = 000000h to 7FFF00h
- DR PV: Memory Area Code = 9Ch, Memory Address for DR0 to DR2 = 000300h to 000500h

## Memory Area Write

### Descripción del comando:

Writes the contents of the specified number of consecutive memory area items starting at the specified address.

### Métodos usados para ejecutar este comando:

Analog Output / Digital Output

### Número de puntos permitidos para este comando:

1-250

### Significado del parámetro P0:

Identifies the CPU Unit connected to the host PC. (See note below).

### Significado del parámetro P1:

Response wait time, in milliseconds (0-150). (See note below).

### Significado del parámetro P2:

Identifies the Destination Network Address of the target PLC (0-127). Use -1 if the target is the directly connected PLC.

### Significado del parámetro P3:

Identifies the Destination Node Address of the target PLC (0-254). Leave empty or 0 if the target is the directly connected PLC. (See note below).

### Significado del parámetro P4:

Identifies the Memory Area Code to be written. (See tables for CS/CJ/CV below).

### Significado del parámetro P5:

Identifies the starting Memory Address to be written. (See tables for CS/CJ/CV below).

### Valores que son enviados:

Valor del punto (0) = First item value to be sent (0-65535)

Valor del punto (1) = Second item value to be sent (0-65535)

...

Valor del punto (n-1) = Last item value to be sent (0-65535)

### NOTE ABOUT DIGITAL OUTPUTS:

When DriverDataType is set to Digital Output, values sent are forced to 0 if value was originally <= 0, or to 1 if value was originally > 0, regardless of the memory Area Code or Item Data Type to be written.

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

## NOTE ABOUT UNIT NUMBER:

The unit number set in DriverP0 is that of the destination CPU Unit connected to the host computer. When the host computer is connected to a CPU Unit, the unit number is designated in the PLC Setup. When the host computer is connected to a Serial Communications Board or a Serial Communications Unit, the unit number is the designated in the Setup for the Board or Unit. DriverP0 accepts values from 0 to 99.

## NOTE ABOUT RESPONSE WAIT TIME:

The response wait time set in DriverP1 sets the time from when the CPU Unit receives a command block until it starts to return a response. It can be set from 0 to 150 ms, in units of 10 ms.

## NOTE ABOUT DESTINATION NODE ADDRESS:

The destination node address is set in DriverP3 and it is required only when sending to a CPU Unit on a network. Set it to -1 if the PLC to be read is the same that is directly connected to the host PC.

*Must be set within the following ranges:*

- PLC directly connected to the Host PC = -1
- Ethernet Units with model numbers ending in ETN21 = 1 to 254
- Ethernet Units with other model numbers = 1 to 126
- Controller Link Unit = 1 to 32
- SYSMAC NET = 1 to 126
- SYSMAC LINK = 1 to 62

## MEMORY ADDRESS TABLE FOR CS/CJ MODE:

Select the data type to be written and set the Memory Area Code in DriverP4 and the Memory Address in DriverP5.

- CIO Bit: Memory Area Code = 30h, Memory Address for CIO 000000 to CIO 614315 = 000000h to 17FF0Fh
- WR Bit: Memory Area Code = 31h, Memory Address for W00000 to W51115 = 000000h to 01FF0Fh
- HR Bit: Memory Area Code = 32h, Memory Address for H00000 to H51115 = 000000h to 01FF0Fh
- AR Bit (read/write): Memory Area Code = 33h, Memory Address for A44800 to A95915 = 01C000h to 03BF0Fh
- CIO Word: Memory Area Code = B0h, Memory Address for CIO 0000 to CIO 6143 = 000000h to 17FF00h
- WR Word: Memory Area Code = B1h, Memory Address for W000 to W511 = 000000h to 01FF00h
- HR Word: Memory Area Code = B2h, Memory Address for H000 to H511 = 000000h to 01FF00h
- AR Word (read /write): Memory Area Code = B3h, Memory Address for A448 to A959 = 01C000h to 03BF00h
- TIM Completion Flag: Memory Area Code = 09h, Memory Address for T0000 to T4095 = 000000h to 0FFF00h
- CNT Completion Flag: Memory Area Code = 09h, Memory Address for C0000 to C4095 = 800000h to 8FFF00h
- TIM PV: Memory Area Code = 89h, Memory Address for T0000 to T4095 = 000000h to 0FFF00h
- CNT PV: Memory Area Code = 89h, Memory Address for C0000 to C4095 = 800000h to 8FFF00h
- DM Bit: Memory Area Code = 02h, Memory Address for D0000000 to D3276715 = 000000h to 7FFF0Fh
- DM Word: Memory Area Code = 82h, Memory Address for D00000 to D32767 = 000000h to 7FFF00h
- EM Bank 0 to Bank F Bit: Memory Area Code = 20h to 2Fh, Memory Address for E0\_0000000 to 3276715 / EF\_0000000 to 3276715 = 000000h to 7FFF0Fh
- EM Bank 10 to Bank 18: Memory Area Code = E0h to E8h, Memory Address for E10\_0000000 to 3276715 / E18\_0000000 0 to 3276715 = 000000h to 7FFF0Fh
- EM Bank 0 to Bank F Word: Memory Area Code = A0h to AFh or 50h to 5Fh, Memory Address for E0\_00000 to 32767 / EF\_00000 to 32767 = 000000h to 7FFF00h
- EM Bank 10 to Bank 18 Word: Memory Area Code = 60h to 68h, Memory Address for E10\_00000 to 32767 / E18\_00000 to 32767 = 000000h to 7FFF00h
- EM Current Bank Bit: Memory Area Code = 0Ah, Memory Address for E0000000 to E3276715 = 000000h to 7FFF0Fh

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012

# Soluciones de comunicación industrial para Windows

- EM Current Bank Word: Memory Area Code = 98h, Memory Address for E00000 to E32767 = 000000h to 7FFF00h
- EM Current Bank No. Word: Memory Area Code = BCh, Memory Address = 0F0000h
- TK Bit: Memory Area Code = 06h, Memory Address for TK0000 to TK0031 = 000000h to 001F00h
- TK Status Bit: Memory Area Code = 46h, Memory Address for TK0000 to TK0031 = 000000h to 001F00h
- IR PV: Memory Area Code = DCh, Memory Address for IR00 to IR15 = 010000h to 010F00h
- DR PV: Memory Area Code = BCh, Memory Address for DR00 to DR15 = 020000h to 020F00h

## MEMORY ADDRESS TABLE FOR CV MODE:

Select the data type to be read and set the Memory Area Code in DriverP4 and the Memory Address in DriverP5.

- CIO Bit: Memory Area Code = 00h, Memory Address for CIO 000000 to CIO 255515 = 000000h to 09FB0Fh
- AR Bit (read/write): Memory Area Code = 00h, Memory Address for A44800 to A95915 = 0CC000h to 0EBF0Fh
- CIO Word: Memory Area Code = 80h, Memory Address for CIO 0000 to CIO 2555 = 000000h to 09FB00h
- AR Word (read /write): Memory Area Code = 80h, Memory Address for A448 to A959 = 0CC000h to 0EBF00h
- TIM Completion Flag: Memory Area Code = 01h, Memory Address for T0000 to T2047 = 000000h to 07FF00h
- CNT Completion Flag: Memory Area Code = 01h, Memory Address for C0000 to C2047 = 080000h to 0FFF00h
- TIM PV: Memory Area Code = 81h, Memory Address for T0000 to T2047 = 000000h to 07FF00h
- CNT PV: Memory Area Code = 81h, Memory Address for C0000 to C2047 = 080000h to 0FFF00h
- DM Word: Memory Area Code = 82h, Memory Address for D00000 to D32767 = 000000h to 7FFF00h
- EM Bank 0 to Bank F Word: Memory Area Code = 90h to 97h, Memory Address for E0\_00000 to 32767 / E7\_00000 to 32767 = 000000h to 7FFF00h
- EM Current Bank Word: Memory Area Code = 98h, Memory Address for E00000 to E32767 = 000000h to 7FFF00h
- DR PV: Memory Area Code = 9Ch, Memory Address for DR0 to DR2 = 000300h to 000500h

## Mensajes de error

La siguiente lista muestra los mensajes de error que pueden ser retornados por el driver en la propiedad 'Status' durante una comunicación fallida.

- [1005] DRIVER (Internal): Invalid driver stage
- [1300] PROTOCOL (Timeout): No answer
- [1433] PROTOCOL (Format): Validation error in device response
- [3007] CONFIG (P0): Invalid device address
- [8348] CONFIG (Remote): Unknown error code
- [8405] CONFIG (Remote): Local node error: Local node not in network
- [8406] CONFIG (Remote): Local node error: Token timeout
- [8407] CONFIG (Remote): Retries failed
- [8408] CONFIG (Remote): Too many send frames
- [8409] CONFIG (Remote): Node address range error
- [8410] CONFIG (Remote): Node address duplication
- [8411] CONFIG (Remote): Destination node error: Destination node not in network
- [8412] CONFIG (Remote): Destination node error: Unit missing
- [8413] CONFIG (Remote): Destination node error: Third node missing
- [8414] CONFIG (Remote): Destination node error: Destination node busy
- [8415] CONFIG (Remote): Destination node error: Response timeout
- [8416] CONFIG (Remote): Controller error: Communications controller error
- [8417] CONFIG (Remote): Controller error: CPU Unit error
- [8418] CONFIG (Remote): Controller error: Controller error
- [8419] CONFIG (Remote): Controller error: Unit number error

CPKSoft Ingeniería  
Drivers para comunicación  
industrial.

www.cpksoft.com.ar  
www.facebook.com/  
cpksoftingenieria  
cpksoftingenieria@  
hotmail.com  
tel: 54-11-1545788354

1990-2012



# Soluciones de comunicación industrial para Windows

[8420] CONFIG (Remote): Service unsupported: Undefined command  
[8421] CONFIG (Remote): Service unsupported: Not supported by model/version  
[8422] CONFIG (Remote): Routing table error: Destination address setting error  
[8423] CONFIG (Remote): Routing table error: No routing tables  
[8424] CONFIG (Remote): Routing table error: Routing table error  
[8425] CONFIG (Remote): Routing table error: Too many relays  
[8426] CONFIG (Remote): Command format error: Command too long  
[8427] CONFIG (Remote): Command format error: Command too short  
[8428] CONFIG (Remote): Command format error: Elements data don't match  
[8429] CONFIG (Remote): Command format error: An incorrect format was used  
[8430] CONFIG (Remote): Command format error: Header error  
[8431] CONFIG (Remote): Parameter error: Area classification missing  
[8432] CONFIG (Remote): Parameter error: Access size error  
[8433] CONFIG (Remote): Parameter error: Address range error  
[8434] CONFIG (Remote): Parameter error: Address range exceeded  
[8435] CONFIG (Remote): Parameter error: Program missing  
[8436] CONFIG (Remote): Parameter error: Relational error  
[8437] CONFIG (Remote): Parameter error: Duplicate data access  
[8438] CONFIG (Remote): Parameter error: Response too long  
[8439] CONFIG (Remote): Parameter error: Parameter error  
[8440] CONFIG (Remote): Read not possible: The program area is protected  
[8441] CONFIG (Remote): Read not possible: Table missing  
[8442] CONFIG (Remote): Read not possible: Data missing  
[8443] CONFIG (Remote): Read not possible: Program missing  
[8444] CONFIG (Remote): Read not possible: File missing  
[8445] CONFIG (Remote): Read not possible: Data mismatch  
[8446] CONFIG (Remote): Write not possible: The specified area is read-only  
[8447] CONFIG (Remote): Write not possible: Protected  
[8448] CONFIG (Remote): Write not possible: Cannot register  
[8449] CONFIG (Remote): Write not possible: Program missing  
[8450] CONFIG (Remote): Write not possible: File missing  
[8451] CONFIG (Remote): Write not possible: File name already exists  
[8452] CONFIG (Remote): Write not possible: Cannot change  
[8453] CONFIG (Remote): Not executable in current mode: Not possible during execution  
[8454] CONFIG (Remote): Not executable in current mode: Not possible while running  
[8455] CONFIG (Remote): Not executable in current mode: Wrong PLC mode (PROGRAM)  
[8456] CONFIG (Remote): Not executable in current mode: Wrong PLC mode (DEBUG)  
[8457] CONFIG (Remote): Not executable in current mode: Wrong PLC mode (MONITOR)  
[8458] CONFIG (Remote): Not executable in current mode: Wrong PLC mode (RUN)  
[8459] CONFIG (Remote): Not executable in current mode: Specified node not polling node  
[8460] CONFIG (Remote): Not executable in current mode: Step cannot be executed  
[8461] CONFIG (Remote): No such device: File device missing  
[8462] CONFIG (Remote): No such device: Memory missing  
[8463] CONFIG (Remote): No such device: Clock missing  
[8464] CONFIG (Remote): Cannot start/stop: Table missing  
[8465] CONFIG (Remote): Unit error: Memory error  
[8466] CONFIG (Remote): Unit error: I/O setting error  
[8467] CONFIG (Remote): Unit error: Too many I/O points  
[8468] CONFIG (Remote): Unit error: CPU bus error  
[8469] CONFIG (Remote): Unit error: I/O duplication  
[8470] CONFIG (Remote): Unit error: I/O bus error  
[8471] CONFIG (Remote): Unit error: SYSMAC BUS/2 error  
[8472] CONFIG (Remote): Unit error: CPU Bus Unit error  
[8473] CONFIG (Remote): Unit error: SYSMAC BUS No. duplication  
[8474] CONFIG (Remote): Unit error: Memory error  
[8475] CONFIG (Remote): Unit error: SYSMAC BUS terminator missing  
[8476] CONFIG (Remote): Command error: No protection  
[8477] CONFIG (Remote): Command error: Incorrect password  
[8478] CONFIG (Remote): Command error: The specified area is protected  
[8479] CONFIG (Remote): Command error: Service already executing  
[8480] CONFIG (Remote): Command error: Service stopped  
[8481] CONFIG (Remote): Command error: No execution right

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/](http://www.facebook.com/)

[cpksoftingenieria@](mailto:cpksoftingenieria@hotmail.com)

[hotmail.com](mailto:cpksoftingenieria@hotmail.com)

tel: 54-11-1545788354

1990-2012



# Soluciones de comunicación industrial para Windows

[8482] CONFIG (Remote): Command error: Settings not complete  
[8483] CONFIG (Remote): Command error: Necessary items not set  
[8484] CONFIG (Remote): Command error: Number already defined  
[8485] CONFIG (Remote): Command error: Error will not clear  
[8486] CONFIG (Remote): Access right error: No access right  
[8487] CONFIG (Remote): Abort: Service aborted

## Equipos soportados

Este driver se puede comunicar con estos equipos, aunque no necesariamente está limitado a los que aparecen en esta lista:

OMRON SYSMAC CS/CJ/CP Series  
OMRON SYSMAC CS1G/H-CPUxxH  
OMRON SYSMAC CS1G/H-CPUxx-EV1  
OMRON SYSMAC CS1D-CPUxxH  
OMRON SYSMAC CS1D-CPUxxS  
OMRON SYSMAC CS1W-SCBxx-V1  
OMRON SYSMAC CS1W-SCUxx-V1  
OMRON SYSMAC CJ2H-CPU6x-EIP  
OMRON SYSMAC CJ2H-CPU6x  
OMRON SYSMAC CJ2M-CPUxx  
OMRON SYSMAC CJ1H-CPUxxH-R  
OMRON SYSMAC CJ1G/H-CPUxxH  
OMRON SYSMAC CJ1G-CPUxxP  
OMRON SYSMAC CJ1G-CPUxx  
OMRON SYSMAC CJ1M-CPUxx  
OMRON SYSMAC CJ1W-SCUxx-V1  
OMRON SYSMAC CP1H-Xxxx-x  
OMRON SYSMAC CP1H-XAxxx-x  
OMRON SYSMAC CP1H-Yxxx-x  
OMRON SYSMAC CP1L-M/Lxxx-x  
OMRON SYSMAC CP1E-ExxDx-x  
OMRON SYSMAC CP1E-NxxDx-x  
OMRON SYSMAC One NSJ Series  
OMRON SYSMAC NSJx-xxxx(B)-G5D  
OMRON SYSMAC NSJx-xxxx(B)-M3D

CPKSoft Ingeniería

Drivers para comunicación industrial.

[www.cpksoft.com.ar](http://www.cpksoft.com.ar)

[www.facebook.com/cpksoftingenieria](http://www.facebook.com/cpksoftingenieria)

[cpksoftingenieria@](mailto:cpksoftingenieria@hotmail.com)

[hotmail.com](mailto:cpksoftingenieria@hotmail.com)

tel: 54-11-1545788354

1990-2012