

# 1 ActiveX RmtX

## 1.1 Generalidades

El componente **RmtX** es un ActiveX que le permite interactuar con dispositivos WebPort Remote desde aplicaciones Windows 32 bits.

### Descripción

Use un componente **RmtX** para establecer comunicaciones entre un WebPort Remote y su aplicación. **RmtX** le provee acceso directo a las entradas y salidas digitales, a los sensores de temperaturas y a los acumuladores del WebPort Remote.

Adicionalmente, mediante el componente **RmtX** se manejan las 3 puertas de comunicaciones del Remote.

Sus propiedades pueden ser especificadas a tiempo de diseño, mediante el inspector de objetos, o a tiempo de ejecución (ver detalles en propiedades, métodos y eventos).

© SageNet 2003

Distribuye Sage SRL  
Heredia 881  
(1427) Ciudad de Buenos Aires  
www.sage.com.ar

## 1.2 Instalacion del componente

El componente ActiveX y sus archivos complementarios se entregan en un cd.

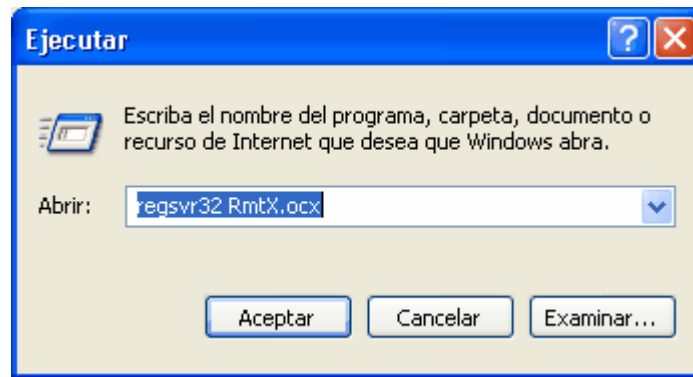
Los archivos son:

**RmtX.ocx** es el componente ActiveX  
**RmtX.hlp** y **RmtX.cnt** son los archivos de ayuda

**PruebaRmt.frm**, **PruebaRmt.vbp** y **PruebaRmt.vbw** ejemplo para Visual Basic 6.0 (véase el capítulo Ejemplo en VB6, para una explicación detallada)

Para la instalación del componente ActiveX, hay que seguir la siguiente secuencia:

1. Copiar el archivo Rmtx.ocx al directorio Windows\System.
2. Copiar los archivos RmtX.hlp y Rmtx.cnt al directorio Windows\Help.
3. Abrir el menú inicio, Ejecutar ...



Tippear "regsvr32 RmtX.ocx", presionar el botón Aceptar (o dar Enter).  
Si el proceso de registro del componente se lleva a cabo con éxito, debe aparecer un mensaje como el de la figura.



A partir de este momento, el componente se halla disponible para ser agregado a la paleta de componentes del Visual Basic. bajo el nombre "Sage WebPort RmtX Library".

## 1.3 Aplicaciones con RmtX

Para incorporar un componente RmtX a su aplicación, deberá seguir las secuencias específicas de acuerdo a la plataforma de desarrollo elegida.

Los 2 casos que se detallan en este capítulo son:

[Visual Basic 6.0](#)

Borland Delphi 4.0

## 1.4 Aplicaciones en Visual Basic 6.0

### Usando el componente RmtX en Microsoft Visual Basic 6.0

#### Introducción

La forma más sencilla de utilizar un componente ActiveX en Visual Basic es arrastrando un componente sobre el form de un proyecto y abriendo el editor de propiedades.

En esta sección, veremos como crear un proyecto simple, arrojando un componente ActiveX RmtX y modificando sus propiedades mediante el editor de propiedades estándar del Visual Basic.

Veremos además como responder a un evento y como modificar el componente con código VB.

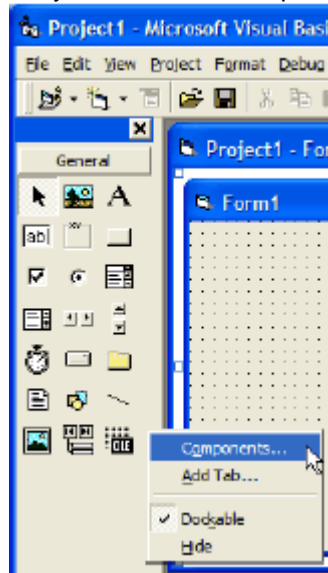
**Nota:** Para llevar a cabo esta secuencia, Ud. deberá haber instalado previamente el componente RmtX en su computadora.

## Agregando componentes a un proyecto

Cada proyecto en Visual Basic se conforma de un conjunto de componentes que serán los que le den su funcionalidad.

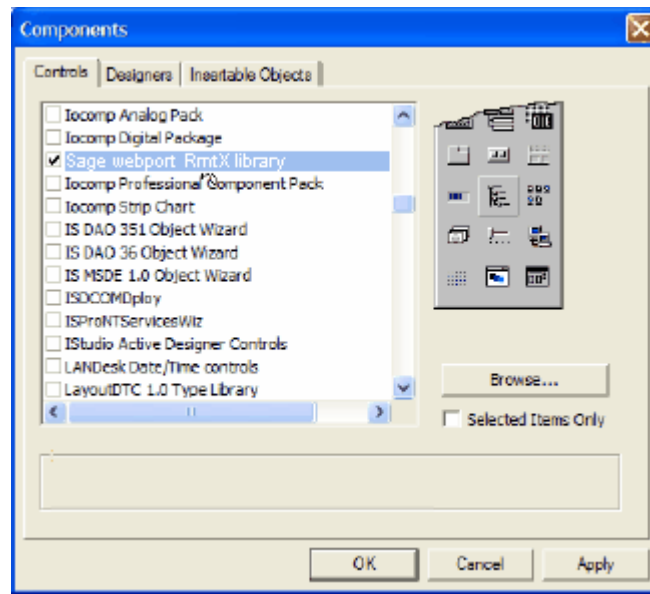
Estos componentes se seleccionan de una lista de componentes disponibles.

Para agregar el componente RmtX a la lista de componentes disponibles (paleta de componentes), hay que hacer click con el botón derecho del mouse (acéptese el anglicismo rightclick para esta acción) sobre la paleta de componentes, y seleccionar "Components..."...



Se presentará una lista con los componentes ActiveX disponibles en su computadora.

Buscando en esta lista, Ud. podrá localizar el componente "Sage WebPort RmtX Library". Colóquele un tilde al casillero de ese componente. Al presionar el botón OK, el componente se agragará a la paleta de componentes disponibles en su proyecto.



### Agregando el componente a un Form

Para agregar un componente RmtX a un form, simplemente haga click sobre el icono respectivo



**RmtX** y arrastrelo sobre el form.

### Editores de propiedades

Seleccionando el componente RmtX, podrán verse y modificarse sus propiedades desde el editor de propiedades del Visual Basic.

Las propiedades accesibles desde el editor de propiedades interno del Visual Basic son solo el IP y la propiedad Activo del RmtX.

Para poder consultar y editar la totalidad de las propiedades del RmtX, se accede mediante un [editor propio](#), que se activa haciendo doble click sobre el componente.

#### Nota importante:

El procedimiento recomendado para modificar las propiedades del RmtX, es mediante código incorporado a la aplicación.

Esto permite documentar mucho mejor el objeto de cada propiedad, y evita el cambio de los valores por equivocaciones desde el ambiente de desarrollo.

Veáse el ejemplo que acompaña al componente.

### Accediendo al componente mediante código

Los cambios que se llevan a cabo mediante el editor de propiedades, se realizan a tiempo de diseño, y quedan almacenados en el proyecto para el momento de su ejecución.

Sin embargo, Ud. necesitará acceder a muchas de las propiedades (así como reaccionar ante los eventos) a tiempo de ejecución, lo que requerirá que acceda mediante código (este es el método sugerido para un mejor diseño del proyecto).

Para acceder a cualquier componente ubicado en un form, lo primero que se necesita es conocer el nombre del componente.

Por defecto, el primer componente RmtX será denominado "RmtX1", pero Ud. puede asignarle un nombre más indicativo, como por ejemplo "WPEmbotelladora" o "RemoteExterior". Tomemos como ejemplo un componente que se ha renombrado como "RemoteExterior"

La forma de acceder a las **propiedades** y **métodos** del componente será de la forma:

```
RemoteExterior.Propiedad  
RemoteExterior.Metodo
```

por ejemplo:

```
RemoteExterior.IP = "10.110.110.200"  
RemoteExterior.Reset
```

y similarmente, se puede acceder a las propiedades y métodos de los objetos contenidos en el componente:

```
RemoteExterior.ComPort(0).Velocidad = 9600  
RemoteExterior.Entrada(6).Descripcion = "Contador de vehículos"
```

## Eventos

Como todo ambiente moderno de desarrollo bajo Windows, las aplicaciones bajo Visual Basic son "por eventos".

Esto significa que el programa se ejecuta reaccionando a eventos, usualmente generados por acciones del usuario (haciendo click sobre un control con el mouse, o mediante el teclado).

Para responder a cada una de estas acciones (eventos), se escriben pequeños tramos de código, llamados manejadores de eventos (Event handlers).

Los manejadores de eventos se activan (se ejecutan) cuando ocurre el evento al que están asociados. Por ejemplo, un evento "OnClick" se ejecuta cuando el usuario hace click con el botón del mouse sobre un componente del form.

Cada componente RmtX posee un conjunto de eventos que están disponibles para permitirle al diseñador de la aplicación generar las respuestas necesarias.

Estos eventos fundamentalmente provienen de cambios que se producen en el WebPort Remote bajo control del ActiveX asociado al mismo.

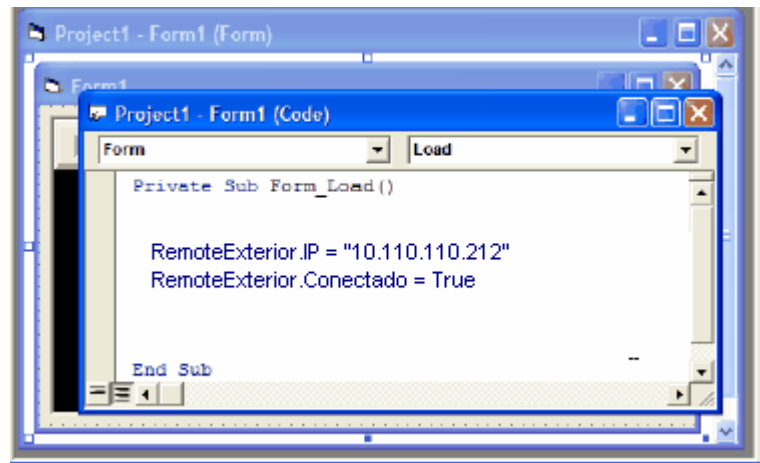
Por ejemplo, cuando se recibe un paquete de datos por la puerta RS232 del Remote, se generará un evento **OnDataIn**, lo que permite al programador tomar las acciones necesarias.

Para crear un manejador de eventos para un evento particular de un componente RmtX, se debe acceder a la ventana de código del Visual Basic.

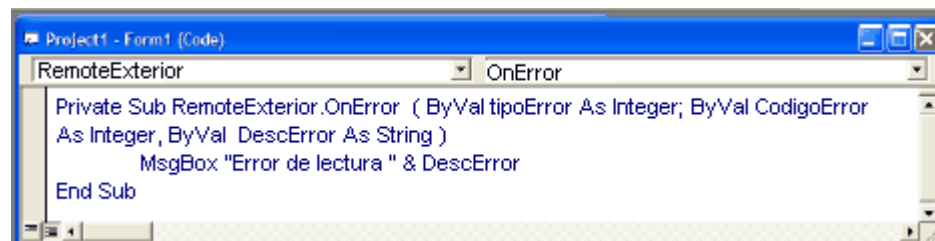
En esa ventana hay dos listas desplegables. La del lado izquierdo contiene una lista de todos los componentes del form (incluido el form mismo).

Un evento especial, que nos permite llevar a cabo funciones de inicialización, es el evento **Form\_Load**, que se ejecuta cuando se crea el form. Es la ubicación correcta para llevar a cabo cualquier función antes de que el form sea visible para el usuario.

En el ejemplo de la figura, se asigna el IP del dispositivo, y se lleva a cabo la conexión.



Para acceder a los eventos específicos del componente RmtX, se selecciona el componente RmtX deseado (en nuestro caso, será el denominado "RemoteExterior") de la lista de la izquierda. A continuación se puede desplegar la lista desplegable disponible a la derecha del componente. Esta lista posee todos los eventos disponibles para el RemoteExterior.



La figura muestra como, al seleccionar el evento OnError del componente RemoteExterior, se accede al tramo de código que representa dicho evento dentro de la aplicación. Algunos eventos (el OnError entre ellos), pasan a la aplicación información adicional (además de avisar que el evento se produjo). Los datos disponibles se describen a continuación del nombre del evento, informando el nombre de la variable asociada, y su tipo.

Por ejemplo:

```
Private Sub RemoteExterior.OnError ( ByVal tipoError As Long; ByVal CodigoError As Long, ByVal DescError As String )
```

Informa un código de tipo de error en forma de una variable entera, un código de origen del error, también como una variable entera, y una descripción del error en forma de una cadena de caracteres.

De este modo, la aplicación podría, por ejemplo generar un mensaje al usuario conteniendo esta información:

```
Private Sub RemoteExterior.OnError ( ByVal tipoError As Long; ByVal CodigoError As Long,
ByVal DescError As String )

    MsgBox "Error en RemoteExterior " & DescError

End Sub
```

Para detalles sobre la sintaxis de las propiedades, métodos y eventos del componente RmtX, veáse [Uso del Componente](#)

## 1.5 Editor de propiedades RmtX

El ActiveX provee su propio editor de propiedades, lo que permite acceder a tiempo de diseño a todas las propiedades del RmtX.

Para acceder a este editor, se debe hacer doble click sobre el componente dentro del form, on lo que se accede a la siguiente ventana:

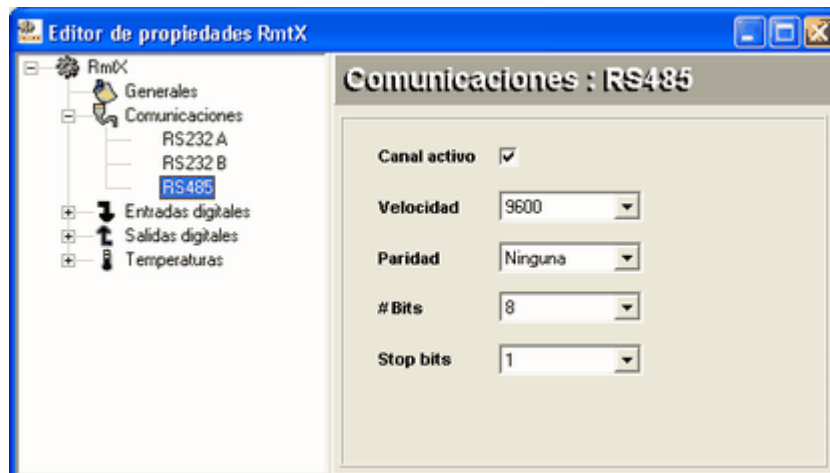


A la izquierda de la ventana se muestra un árbol con el conjunto de propiedades accesibles desde el editor.

Para expandir o contraer una rama del árbol, se hace click sobre el recuadro con el signo (+) ó (-), o bien se puede hacer doble click sobre el nombre de la rama.

A la derecha se encuentra la zona de edición de propiedades, la que solo permite editar cuando se ha seleccionado un elemento específico del componente.

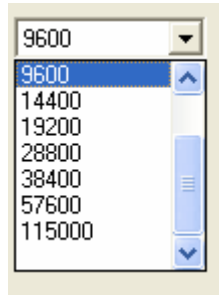
En la siguiente figura, como ejemplo, se ha seleccionado el canal RS485:



En la ventana de edición se muestran los valores actuales de las propiedades de cada canal RS485 (Canal activo, velocidad, paridad, bits y stop bits), y el usuario puede modificarlas utilizando para ello los campos específicos.

En todos los casos en que la selección esté limitada a un subconjunto de valores posibles, el editor presentará una lista desplegable, lo que facilita la selección y evita errores (en el ejemplo la lista

desplegable de selección de velocidades):



Navegando por el árbol, se accede a cada una de las páginas de propiedades. Al cerrar la ventana, las nuevas definiciones quedan activas, y son almacenadas en el proyecto.

Por tratarse se un editor que funciona a tiempo de diseño, no se lleva a cabo ninguna comunicación con el WebPort Remote, de modo que todos los valores aquí visibles son almacenados dentro del proyecto.

## 1.6 Uso del Componente

### 1.6.1 Generalidades

El ActiveX RmtX permite implementar una aplicación bajo entorno Windows de 32 bits para acceder a todas las funciones del WebPort Remote<sup>®</sup>.

A continuación se describen las propiedades, los eventos y los métodos de este componente, agrupados de acuerdo a las funciones del Remote a las que hacen referencia.

En todos los casos se describe la sintaxis para aplicaciones en Visual Basic y en Delphi.

Funciones generales

Comunicaciones seriales

Entradas digitales

Salidas digitales

Temperaturas

Registros digitales

### 1.6.2 Funciones generales

#### Propiedad Activo

Permite activar y desactivar (o bien controlar el estado de actividad) del WebPort Remote.

Cuando el Remote está inactivo, el mismo deja de actualizar las propiedades del ActiveX, y deja de generar eventos.

Similarmente, los cambios a las propiedades con capacidad de escritura generarán un error.

Al reactivarse (Activo=true), el ActiveX vuelve a reflejar el estado del dispositivo.

Adicionalmente, al pasar de False a True esta propiedad, el ActiveX se comunica con el WebPort Remote, y lee los valores por defecto de todas las propiedades.  
Por último, al pasar a activo, el ActiveX pone en funcionamiento sus timers internos, los que llevan a cabo los procesos automáticos de lectura.

### Sintaxis

Delphi	myBoolean := Objeto. <b>Activo</b> ; Objeto. <b>Activo</b> := myBoolean;
Visual Basic	myBoolean = Objeto. <b>Activo</b> Objeto. <b>Activo</b> = myBoolean

### Modo

Lectura/escritura.

### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

### Valores Posibles

True:	Activo.
False:	Inactivo.

## Propiedad IP

Es la dirección IP del WebPort Remote asociado al control ActiveX.  
Identifica unívocamente a un dispositivo WebPort Remote dentro de la red de controladores.

### Sintaxis

Delphi	myString := Objeto. <b>IP</b> ; Objeto. <b>IP</b> := myString;
Visual Basic	myString = Objeto. <b>IP</b> Objeto. <b>IP</b> = myString

### Modo

Lectura/escritura.

### Tipo de dato

Delphi	String
Visual Basic	String

### Valores

Hasta 15 caracteres (nnn.nnn.nnn.nnn).

## Propiedad TiempoDeMarcha

Es el tiempo de funcionamiento del Remote desde el último encendido.

### Sintaxis

Delphi	myDateTime := Objeto. <b>TiempoDeMarcha</b> ;
Visual Basic	myDate = Objeto. <b>TiempoDeMarcha</b>

### Modo

Solo lectura.

### Tipo de dato

Delphi	tDateTime
Visual Basic	Date

### Propiedad Contacto

Es el nombre de la persona responsable de administrar este dispositivo. Adicionalmente puede contener información acerca de como contactar a dicha persona.

#### Sintaxis

Delphi	myString := Objeto. <b>Contacto</b> ;
Visual Basic	myString = Objeto. <b>Contacto</b>

#### Modo

Solo lectura.

#### Tipo de dato

Delphi	String
Visual Basic	String

#### Valores

Hasta 256 caracteres.

### Propiedad Identificacion

Es la identificación del WebPort Remote asociado a este ActiveX.

#### Sintaxis

Delphi	myString := Objeto. <b>Identificacion</b> ;
Visual Basic	myString = Objeto. <b>Identificacion</b>

#### Modo

Solo lectura

#### Tipo de dato

Delphi	String
Visual Basic	String

#### Valores

Hasta 256 caracteres.

### Propiedad Ubicacion

Permite especificar la ubicación física del WebPort Remote.

#### Sintaxis

Delphi	myString := Objeto. <b>Ubicación</b> ;
Visual Basic	myString = Objeto. <b>Ubicación</b>

#### Modo

Solo lectura.

#### Tipo de dato

Delphi	String
Visual Basic	String

## Valores

Hasta 256 caracteres.

## Propiedad ModoES

Permite leer la configuración de canales digitales del WebPort Remote asociado.

El WebPort puede ser configurado con 8 canales de entrada y 8 canales de salida, o bien con 16 canales de entrada.

## Sintaxis

Delphi                      myInteger := Objeto.**ModoES**;  
Visual Basic              myLong = Objeto.**ModoES**

## Modo

Solo lectura.

## Tipo de dato

Delphi                      Integer  
Visual Basic              Long

## Valores

0: Configurado como 16 entradas.

1: Configurado como 8 entradas más 8 salidas.

## Propiedad Entradas

Permite leer el estado de las 16 entradas digitales, en forma conjunta.

## Sintaxis

Delphi                      myInteger := Objeto.**Entradas**;  
Visual Basic              myLong = Objeto.**Entradas**

## Modo

Solo lectura.

## Tipo de dato

Delphi                      Integer  
Visual Basic              Long

## Valores

El entero resultante de esta propiedad contiene bit a bit el estado de las 16 entradas.

## Propiedad FechaHora

Permite leer y ajustar la fecha y hora del reloj de tiempo real del WebPort Remote.

En caso de pérdida de alimentación, el WebPort Remote pierde la fecha y la hora. En ese caso, puede utilizarse esta propiedad para reconfigurar el dispositivo en base a la fecha y hora de la PC donde se ejecuta la aplicación.

## Sintaxis

Delphi                      myDateTime := Objeto.**FechaHora**;  
Objeto.**FechaHora** := myDateTime;  
Visual Basic              myDate = Objeto.**FechaHora**  
Objeto.**FechaHora** = myDate

## Modo

Solo lectura.

**Tipo de dato**

Delphi	tDateTime
Visual Basic	Date

**Propiedad VersionRmtX**

Permite leer el número de versión del componente ActiveX implementado.

**Sintaxis**

Delphi	myString := Objeto. <b>VersionRmtX</b> ;
Visual Basic	myString = Objeto. <b>VersionRmtX</b>

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	String
Visual Basic	String

**Propiedad VersionFirmware**

Permite leer el número de versión del firmware interno del WebPort Remote conectado al componente RmtX.

**Sintaxis**

Delphi	myString := Objeto. <b>VersionFirmware</b> ;
Visual Basic	myString = Objeto. <b>VersionFirmware</b>

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	String
Visual Basic	String

**Propiedades de rangos disponibles**

Hay 3 propiedades especiales del RmtX, que proveen el rango disponible de entradas, salidas y sensores.

Los valores de estas propiedades reflejan el tipo de hardware y la programación interna del WebPort Remote.

Para la versión actual del firmware, las propiedades y sus valores son:

**Propiedad MAX\_ENTRADAS\_DIG**

Es la cantidad de entyradas digitales configuradas.

**Sintaxis**

Delphi	myInteger:= Objeto. <b>MAX_ENTRADAS_DIG</b> ;
Visual Basic	myLong = Objeto. <b>MAX_ENTRADAS_DIG</b>

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Integer
Visual Basic	Long

Valor por defecto      **16**

**Propiedad MAX\_SENSORES\_TEMP**

Es la cantidad de sensores de temperatura.

**Sintaxis**

Delphi	myInteger:= Objeto.MAX_SENSORES_TEMP;
Visual Basic	myLong = Objeto.MAX_SENSORES_TEMP

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Integer
Visual Basic	Long

Valor por defecto      **4**

**Propiedad MAX\_SALIDAS\_DIG**

Cantidad de salidas digitales (El remote puede estar configurado para 8 ó 0).

**Sintaxis**

Delphi	myInteger:= Objeto.MAX_SALIDAS_DIG;
Visual Basic	myLong = Objeto.MAX_SALIDAS_DIG

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Integer
Visual Basic	Long

Valor por defecto      **0**

**Propiedad MAX\_COM\_PORTS**

Cantidad de poertas de comunicaciones habilitadas.

**Sintaxis**

Delphi	myInteger:= Objeto.MAX_COM_PORTS;
Visual Basic	myLong = Objeto.MAX_COM_PORTS

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Integer
Visual Basic	Long

Valor por defecto      **3**



### Valores Posibles

True: Activo.  
False: Inactivo.

### Propiedad Velocidad

Permite establecer la velocidad a la que se comunican cada una de las 3 puertas de comunicaciones del Remote.

#### Sintaxis

Delphi Objeto.**ComPort[0].Velocidad** := myInteger;  
Visual Basic Objeto.**ComPort(0).Velocidad** = myLong  
(El índice 0 indica puerto RS232 A).

#### Modo

Lectura/escritura.

#### Tipo de dato

Delphi Integer  
Visual Basic Long

#### Valores posibles

1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 y 115000.  
Valor por defecto **9600**.

**Nota:** ante un reset del componente, este parámetro retorna a su valor por defecto.

### Propiedad Paridad

Permite establecer la paridad para cada una de las 3 puertas de comunicaciones del Remote.

#### Sintaxis

Delphi Objeto.**ComPort[0].Paridad** := myInteger;  
Visual Basic Objeto.**ComPort(0).Paridad** = myLong  
(El índice 0 indica puerto RS232 A).

#### Modo

Lectura/escritura.

#### Tipo de dato

Delphi Integer  
Visual Basic Long

#### Valores posibles

0: Ninguna  
1: Impar  
2: Par  
Valor por defecto **0 (ninguna)**.

**Nota:** ante un reset del componente, este parámetro retorna a su valor por defecto.

### Propiedad Bits

Permite establecer la cantidad de bits para cada una de las 3 puertas de comunicaciones del Remote.

#### Sintaxis

Delphi                      Objeto.**ComPort[0].Bits** := myInteger;  
 Visual Basic              Objeto.**ComPort(0).Bits** = myLong  
 (El índice 0 indica puerto RS232 A).

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi                      Integer  
 Visual Basic              Long

**Valores posibles**

7, 8.  
 Valor por defecto        **8.**

**Nota:** ante un reset del componente, este parámetro retorna a su valor por defecto.

**Propiedad StopBits**

Permite establecer la cantidad de StopBits para cada una de las 3 puertas de comunicaciones del Remote.

**Sintaxis**

Delphi                      Objeto.**ComPort[0].StopBits** := myInteger;  
 Visual Basic              Objeto.**ComPort(0).StopBits** = myLong  
 (El índice 0 indica puerto RS232 A).

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi                      Integer  
 Visual Basic              Long

**Valores posibles**

1, 2.  
 Valor por defecto        **1.**

**Nota:** ante un reset del componente, este parámetro retorna a su valor por defecto.

**Método PutString**

Transmite una cadena de caracteres por uno de los canales seriales.  
 El número de canal serial se pasa como parámetro.  
 Da como resultado un código de estado.

**Sintaxis**

Delphi                      Function Objeto.**PutString**(nroCanal: Integer, myString: String): Integer;  
 Visual Basic              Function Objeto.**PutString**(nroCanal as Long, myString as String) As Long

**Argumentos**

Delphi                      nroCanal: Integer  
                                  myString: String  
 Visual Basic

nroCanal: Long  
myString: String

### Resultado

Se obtiene un código de estado:

0: éxito.  
-1: comando inválido.  
-2: timeout.  
-3: dirección inválida.  
-4: error de comunicaciones.

### Método PutChar

Transmite un caracter por uno de los canales seriales.

El número de canal serial se pasa como parámetro.

Dá como resultado un código de estado.

### Sintaxis

Delphi Function Objeto.**PutChar**(nroCanal: Integer, myChar: Char): Integer;  
Visual Basic Function Objeto.**PutChar**(ByVal nroCanal as Long, ByVal myChar as Char)  
As Long

### Argumentos

Delphi  
nroCanal: Integer  
myChar: Char

Visual Basic  
nroCanal: Long  
myChar: Char

### Resultado

Se obtiene un código de estado:

0: éxito.  
-1: comando inválido.  
-2: timeout.  
-3: dirección inválida.  
-4: error de comunicaciones.

### Método GetChar

Recibe un caracter desde uno de los canales seriales.

El número de canal serial se pasa como parámetro.

Dá como resultado un código de estado.

### Sintaxis

Delphi Function Objeto.**GetChar**(nroCanal: Integer): SmallInt;  
Visual Basic Function Objeto.**GetChar**(ByVal nroCanal as Long) As Int

### Argumentos

Delphi  
nroCanal: Integer

Visual Basic  
nroCanal: Long

### Resultado

Devuelve el código ASCII del caracter recibido.

### Método CharReady

Permite conocer si existe un caracter en el buffer de comunicaciones del canal indicado. Dá como resultado True o False.

#### Sintaxis

Delphi                      Function Objeto.**CharReady**(nroCanal: Integer): Boolean;  
 Visual Basic              Function Objeto.**CharReady**(ByVal nroCanal as Long) As Boolean

#### Argumentos

Delphi                      nroCanal: Integer  
 Visual Basic              nroCanal: Long

#### Resultado

Devuelve True si hay caracteres en el buffer del canal.

### Evento OnDataIn

Cuando se reciben datos en alguno de los canales seriales, se genera un evento OnDataIn. Los datos recibidos se acumulan en una cadena de caracteres que se pasa como parámetro. Dos parámetros adicionales indican el número del canal receptor, y la cantidad de caracteres leídos.

#### Sintaxis

Delphi                      Sub Objeto.**OnDataIn**(Canal: integer; Datos: string; Cuenta: integer);  
 Visual Basic              Sub Objeto.**OnDataIn**(ByVal Canal: Long, ByVal Datos: string, ByVal Cuenta:  
 Long)

#### Argumentos

Delphi                      Canal: Integer, indica el canal receptor  
                                  Datos: String, contiene los datos recibidos por el canal serie especificado.  
                                  Cuenta: Integer, mantiene la cuenta de la cantidad de caracteres recibidos.  
 Visual Basic              Canal: Long, indica el canal receptor  
                                  Datos: String, contiene los datos recibidos por el canal serie especificado.  
                                  Cuenta: Long, mantiene la cuenta de la cantidad de caracteres recibidos.

## 1.6.4 Entradas digitales

Las entradas digitales se implementan mediante la propiedad **Entrada**.

**Entrada** debe referenciar a una de las 16 entradas digitales disponibles en el WebPort Remote (8 entradas si el WebPort está configurado como 8 entradas y 8 salidas).

Los índices para referirse a las entradas (y como regla general, en todas los casos en que se utilizan índices), comienzan en 0 (cero).

De este modo las entradas digitales se enumeran:

0, 1, ... , 15 (si se trata de un Remote con 16 entradas).

0, 1, .. 7 (para el caso de 8 entradas).

### Propiedad Descripcion

Es un texto asociado a cada entrada digital.

#### Sintaxis

Delphi                      Objeto.**Entrada[0].Descripcion** := myString;  
                                  myString := Objeto.**Entrada[0].Descripcion**;

Visual Basic      Objeto.**Entrada(0).Descripcion** = myString  
myString = Objeto.**Entrada(0).Descripcion**  
(El índice 0 indica el primer canal digital)

**Modo**

Lectura/escritura

**Tipo de dato**

Delphi                      String  
Visual Basic              String

**Valores**

Hasta 256 caracteres.

**Propiedad DescripcionAlarma**

Es el texto que se asociará a la alarma para cada entrada digital.

**Sintaxis**

Delphi                      Objeto.**Entrada[0].DescripcionAlarma** := myString;  
myString := Objeto.**Entrada[0].DescripcionAlarma**;  
Visual Basic              Objeto.**Entrada(0).DescripcionAlarma** = myString  
myString = Objeto.**Entrada(0).DescripcionAlarma**  
(El índice 0 indica el primer canal digital)

**Modo**

Lectura/escritura

**Tipo de dato**

Delphi                      String  
Visual Basic              String

**Valores**

Hasta 256 caracteres.

**Propiedad Habilitada**

Cuando una entrada digital se encuentra habilitada, la misma es controlada por el WebPort Remote, generano una alarma cada vez que pasa del estado habitual (ver propiedad EstadoBase), al estado complementario.

**Sintaxis**

Delphi                      myBoolean := Objeto.**Entrada[0].Habilitada**;  
Objeto.**Entrada[0].Habilitada** := myBoolean;  
Visual Basic              myBoolean = Objeto.**Entrada(0).Habilitada**  
Objeto.**Entrada(0).Habilitada** = myBoolean  
(El índice 0 indica el primer canal digital)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi                      Boolean  
Visual Basic              Boolean

**Valores Posibles**

True: Entrada Habilitada.  
False: Entrada Deshabilitada.

### Propiedad EstadoNormal

Cada canal posee un estado normal.

Cuando el canal pasa de este estado al estado complementario, se generará una alarma (siempre que la propiedad "**Habilitada**" de dicho canal esté en 1).

#### Sintaxis

Delphi                    myBoolean := Objeto.**Entrada[0].EstadoNormal**;  
Objeto.**Entrada[0].EstadoNormal** := myBoolean;  
Visual Basic            myBoolean = Objeto.**Entrada(0).EstadoNormal**  
Objeto.**Entrada(0).EstadoNormal** = myBoolean  
(El índice 0 indica el primer canal digital)

#### Modo

Lectura/escritura.

#### Tipo de dato

Delphi                    Boolean  
Visual Basic            Boolean

#### Valores Posibles

True:                    Normalmente Activo (esto significa que usualmente hay tensión en esa entrada).  
False:                   Normalmente Inactivo (usualmente no hay tensión aplicada a esa entrada).

### Propiedad EstadoActual

Refleja el estado instantáneo de cada entrada digital.

#### Sintaxis

Delphi                    myBoolean := Objeto.**Entrada[0].EstadoActual**;  
Visual Basic            myBoolean = Objeto.**Entrada(0).EstadoActual**  
(El índice 0 indica el primer canal digital)

#### Modo

Solo lectura.

#### Tipo de dato

Delphi                    Boolean  
Visual Basic            Boolean

#### Valores Posibles

True:                    Activo (hay tensión en esa entrada).  
False:                   Inactivo (no hay tensión aplicada a esa entrada).

### Propiedad Contador

Cada canal digital tiene asociado un contador interno, que se incrementa cada vez que el estado pasa de EstadoNormal al estado complementario (pasa de 1 a 0 o bien de 0 a 1 según cual sea el estado normal de la entrada).

Los contadores reflejan cambios de estado de duración mayor o igual a 1 milisegundo.

**Sintaxis**

Delphi `myInteger := Objeto.Entrada[0].Contador;  
Objeto.Entrada[0].Contador := myInteger;`

Visual Basic `myLong = Objeto.Entrada(0).Contador  
Objeto.Entrada(0).EstadoBase = myLong  
(El índice 0 indica el primer canal digital)`

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi Integer

Visual Basic Long

**Valores Posibles**

-2147483648..2147483647 ( 32 bits, con signo).

**Propiedad Maximo**

Si el canal digital se utiliza como acumulador, podrá generarse un evento cuando el mismo alcance el valor ingresado en esta propiedad.

El evento a ser generado es **OnContadorMaximoAlcanzado**.

Si el valor de esta propiedad es nulo (=0), no se generará nunca el evento citado.

**Sintaxis**

Delphi `myInteger := Objeto.Entrada[0].Maximo;  
Objeto.Entrada[0].Maximo := myInteger;`

Visual Basic `myLong = Objeto.Entrada(0).Maximo  
Objeto.Entrada(0).Maximo = myLong  
(El índice 0 indica el primer canal digital)`

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi Integer

Visual Basic Long

**Valores Posibles**

0 .. 2147483647

**Propiedad PreMaximo**

Si el canal digital se utiliza como acumulador, puede ingresarse un valor previo al máximo (lo denominamos PreMaximo), de modo de habilitar el evento **PreMaximoAlcanzado**, lo que permitirá llevar a cabo alguna acción a la aplicación.

Si este valor está en cero, no se generará nunca el evento citado.

**Sintaxis**

Delphi `myInteger := Objeto.Entrada[0].PreMaximo;  
Objeto.Entrada[0].PreMaximo := myInteger;`

Visual Basic `myLong = Objeto.Entrada(0).PreMaximo  
Objeto.Entrada(0).PreMaximo = myLong  
(El índice 0 indica el primer canal digital)`

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi	Integer
Visual Basic	Long

**Valores Posibles**

0 .. 2147483647

**Propiedad GeneraAlarmas**

Si se desea la generación de una alarma cuando un canal digital pasa de su estado normal al complementario, debe indicarse esta propiedad como True.

**Sintaxis**

Delphi	myBoolean := Objeto. <b>Entrada[0].GeneraAlarmas</b> ; Objeto. <b>Entrada[0].GeneraAlarmas</b> := myBoolean;
Visual Basic	myBoolean = Objeto. <b>Entrada(0).GeneraAlarmas</b> Objeto. <b>Entrada(0).GeneraAlarmas</b> = myBoolean (El índice 0 indica el primer canal digital)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi	Boolean
Visual Basic	Boolean

**Valores Posibles**

True:	Alarmas de la entrada digital activadas.
False:	Alarmas de la entrada digital desactivadas.

**Propiedad ResetEnMaximo**

Si se desea que una vez alcanzado el máximo, el acumulador respectivo retorne a cero automáticamente, deberá establecerse esta propiedad en True.

**Sintaxis**

Delphi	myBoolean := Objeto. <b>Entrada[0].ResetEnMaximo</b> ); Objeto. <b>Entrada[0].ResetEnMaximo</b> := myBoolean;
Visual Basic	myBoolean = Objeto. <b>Entrada(0).ResetEnMaximo</b> Objeto. <b>Entrada(0).ResetEnMaximo</b> = myBoolean (El índice 0 indica el primer canal digital)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi	Boolean
Visual Basic	Boolean

**Valores Posibles**

True:	Reset activado.
False:	Reset desactivado.

## Propiedad IntervaloMuestreo

Esta propiedad determina el intervalo de interrogación al WebPort Remote para chequear el estado de una entrada digital.

Si bien el Remote mantiene el estado cuasi-instantáneo de sus entradas, la aplicación puede ajustar el tiempo de actualización de esta información, lo que determinará la generación del evento

### OnAlarmaDigital.

Las unidades de este intervalo son décimas de segundo.

Si este valor está en cero, no se generará nunca el evento citado.

### Sintaxis

Delphi	myInteger := Objeto. <b>Entrada[0].IntervaloMuestreo</b> ; Objeto. <b>Entrada[0].IntervaloMuestreo</b> := myInteger;
Visual Basic	myLong = Objeto. <b>Entrada(0).IntervaloMuestreo</b> Objeto. <b>Entrada(0).IntervaloMuestreo</b> = myLong (El índice 0 indica el primer canal digital)

### Modo

Lectura/escritura.

### Tipo de dato

Delphi	Integer
Visual Basic	Long

### Valores Posibles

0 .. 2147483647

Valor por defecto: **5 décimas de segundo.**

## Evento OnAlarmaDigital

Cuando una entrada digital está habilitada, y pasa del estado definido en **EstadoNormal** al estado complementario (pasa de 0 a 1 ó de 1 a 0 según corresponda), se genera un evento

### OnAlarmaDigital.

Este evento pasa como parámetro el número de canal en el que se produjo la alarma.

### Sintaxis

Delphi	Sub Objeto. <b>OnAlarmaDigital</b> (Canal: integer);
Visual Basic	Sub Objeto. <b>OnAlarmaDigital</b> (ByVal Canal: Long)

### Argumento

Delphi 4	Canal: Integer, indica el canal en alarma.
Visual Basic	Canal: Long, indica el canal en alarma.

## Evento OnContadorMaximoAlcanzado

Cuando un contador alcanza el valor especificado en **Maximo**, se genera un evento

### OnContadorMaximoAlcanzado.

### Sintaxis

Delphi	Sub Objeto. <b>OnContadorMaximoAlcanzado</b> (Canal: integer);
Visual Basic	Sub Objeto. <b>OnContadorMaximoAlcanzado</b> (ByVal Canal: Long)

### Argumento

Delphi 4	Canal: Integer, indica el canal con máximo alcanzado.
Visual Basic	Canal: Long, indica el canal con máximo alcanzado.

### Evento **OnContadorPreMaximoAlcanzado**

Cuando un contador alcanza el valor especificado en **PreMaximo**, se genera un evento **OnContadorPreMaximoAlcanzado**.

#### Sintaxis

Delphi                      Sub Objeto.**OnContadorPreMaximoAlcanzado**(Canal: integer);  
Visual Basic              Sub Objeto.**OnContadorPreMaximoAlcanzado**(ByVal Canal: Long)

#### Argumento

Delphi 4                      Canal: Integer, indica el canal con máximo alcanzado.  
Visual Basic                Canal: Long, indica el canal con máximo alcanzado.

### Evento **OnRetornoDigitalNormal**

Cuando una entrada digital está habilitada, y retorna al **EstadoNormal** luego de haber pasado por un estado de alarma, se genera un evento **OnRetornoDigitalNormal**.

#### Sintaxis

Delphi                      Sub Objeto.**OnRetornoDigitalNormal**(Canal: integer);  
Visual Basic              Sub Objeto.**OnRetornoDigitalNormal**(ByVal Canal: Long)

#### Argumento

Delphi 4                      Canal: Integer, indica el canal con máximo alcanzado.  
Visual Basic                Canal: Long, indica el canal con máximo alcanzado.

## 1.6.5 Salidas digitales

Las salidas digitales se implementan mediante la propiedad **Salida**.

**Salida** debe referenciar a una de las 8 salidas digitales disponibles en el WebPort Remote.

Los índices para referirse a las salidas (y como regla general, en todos los casos en que se utilizan índices), comienzan en 0 (cero).

De este modo las salidas digitales se enumeran:

0, 1, .. 7

### Propiedad **Descripcion**

Es un texto asociado a cada salida digital.

#### Sintaxis

Delphi	Objeto. <b>Salida[0].Descripcion</b> := myString; myString := Objeto. <b>Salida[0].Descripcion</b> ;
Visual Basic	Objeto. <b>Salida(0).Descripcion</b> = myString myString = Objeto. <b>Salida(0).Descripcion</b> (El índice 0 indica el primer canal digital)

#### Modo

Lectura/escritura

#### Tipo de dato

Delphi	String
Visual Basic	String

#### Valores

Hasta 256 caracteres.

### Propiedad **EstadoBase**

Cada salida digital posee estado base, que representa el estado en el que se mantiene por defecto.

El WebPort Remote coloca cada salida en este estado ante una reinicialización (por encendido o por el método **Reset**).

#### Sintaxis

Delphi	myBoolean := Objeto. <b>Salida[0].EstadoBase</b> ; Objeto. <b>Salida[0].EstadoBase</b> := myBoolean;
Visual Basic	myBoolean = Objeto. <b>Salida(0).EstadoBase</b> Objeto. <b>Salida(0).EstadoBase</b> = myBoolean (El índice 0 indica el primer canal digital)

#### Modo

Lectura/escritura.

#### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

#### Valores Posibles

True:	Normalmente Activo (se comporta como una llave cerrada).
False:	Normalmente Inactivo (es una llave abierta).

## Propiedad EstadoActual

Determina y refleja el estado instantáneo de cada salida digital.

### Sintaxis

Delphi	<code>myBoolean := Objeto.<b>Salida[0].EstadoActual</b>;</code> <code>Objeto.<b>Salida[0].EstadoActual</b> := myBoolean;</code>
Visual Basic	<code>myBoolean = Objeto.<b>Salida(0).EstadoActual</b></code> <code>Objeto.<b>Salida(0).EstadoActual</b> = myBoolean</code> (El índice 0 indica el primer canal digital)

### Modo

Lectura/escritura.

### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

### Valores Posibles

True:	Salida activa (contacto cerrado).
False:	Salida Inactiva (contacto abierto).

## 1.6.6 Temperaturas

Los canales de toma de temperaturas se implementan mediante la propiedad **Temperatura**. **Temperatura** debe referenciar a uno de los 4 canales de sensado disponibles en el WebPort Remote. Los índices para referirse a las temperaturas (y como regla general, en todos los casos en que se utilizan índices), comienzan en 0 (cero).

De este modo las temperaturas se enumeran:  
0, 1, 2 y 3.

### Propiedad Habilitado

Esta propiedad refleja si se procesan las mediciones y los eventos del canal respectivo.

#### Sintaxis

Delphi	<code>Objeto.Temperatura[0].Habilitado := myBoolean;</code> <code>myBoolean := Objeto.Temperatura[0].Habilitado;</code>
Visual Basic	<code>Objeto.Temperatura(0).Habilitado = myBoolean</code> <code>myBoolean = Objeto.Temperatura(0).Habilitado</code> (El índice 0 indica el primer canal de medición de temperaturas)

#### Modo

Lectura/escritura.

#### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

#### Valores Posibles

True:	Sensado Habilitado.
False:	Sensado Deshabilitado.

### Propiedad Descripcion

Es un texto asociado a cada canal de toma de temperaturas.

#### Sintaxis

Delphi	<code>Objeto.Temperatura[0].Descripcion := myString;</code> <code>myString := Objeto.Temperatura[0].Descripcion;</code>
Visual Basic	<code>Objeto.Temperatura(0).Descripcion = myString</code> <code>myString = Objeto.Temperatura(0).Descripcion</code> (El índice 0 indica el primer canal de medición de temperatura),

#### Modo

Lectura/escritura

#### Tipo de dato

Delphi	String
Visual Basic	String

#### Valores posibles

Hasta 256 caracteres.

### Propiedad DescripcionAlarma

Es el texto que se asociará a la alarma para cada entrada de sensado de temperatura.

### Sintaxis

Delphi	Objeto. <b>Temperatura[0].DescripcionAlarma</b> := myString; myString := Objeto. <b>Temperatura[0].DescripcionAlarma</b> ;
Visual Basic	Objeto. <b>Temperatura(0).DescripcionAlarma</b> = myString myString = Objeto. <b>Temperatura(0).DescripcionAlarma</b> (El índice 0 indica el primer canal de medición de temperaturas)

### Modo

Lectura/escritura

### Tipo de dato

Delphi	String
Visual Basic	String

### Valores posibles

Hasta 256 caracteres.

## Propiedad Conectado

Esta propiedad refleja si se encuentra conectado un termómetro al canal respectivo.

### Sintaxis

Delphi	myBoolean := Objeto. <b>Temperatura[0].Conectado</b> ;
Visual Basic	myBoolean = Objeto. <b>Temperatura(0).Conectado</b> (El índice 0 indica el primer canal de medición de temperaturas)

### Modo

Solo lectura.

### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

### Valores Posibles

True:	Sensor conectado.
False:	Sensor desconectado.

## Propiedad IdTermometro

Cada termómetro digital tiene grabado un número interno único, que permite identificarlo dentro de una red de sensores.

### Sintaxis

Delphi	myString := Objeto. <b>Temperatura[0].IdTermometro</b> ;
Visual Basic	myString = Objeto. <b>Temperatura(0).IdTermometro</b> (El índice 0 indica el primer canal de medición de temperaturas),

### Modo

Solo lectura.

### Tipo de dato

Delphi	String
Visual Basic	String

### Valores posibles

Hasta 256 caracteres.

### Propiedad ValorActual

Refleja el valor actual del sensor de temperatura asociado a este canal.

#### Sintaxis

Delphi `myReal := Objeto.Temperatura[0].ValorActual;`  
Visual Basic `myFloat = Objeto.Temperatura(0).ValorActual`  
(El índice 0 indica el primer canal de medición de temperaturas)

#### Modo

Solo lectura.

#### Tipo de dato

Delphi Real  
Visual Basic Float

#### Valores Posibles

MIN\_TEMP\_VALUE .. MAX\_TEMP\_VALUE (ver constantes)

### Propiedad ValorMaximo

Si esta propiedad tiene un valor no nulo, el RmtX generará un evento **OnAlarmaTemp** cuando la medición lo supere.

#### Sintaxis

Delphi `myReal := Objeto.Temperatura[0].ValorMaximo;`  
Visual Basic `myFloat = Objeto.Temperatura(0).ValorMaximo`  
(El índice 0 indica el primer canal de medición de temperaturas)

#### Modo

Lectura/escritura.

#### Tipo de dato

Delphi Real  
Visual Basic Float

#### Valores Posibles

MIN\_TEMP\_VALUE .. MAX\_TEMP\_VALUE (ver constantes)

### Propiedad ValorPreMaximo

Si esta propiedad tiene un valor no nulo, el RmtX generará un evento **OnPreAlarmaTemp** cuando la medición de temperaturas se encuentre por encima de este valor y por debajo del ValorMaximo.

#### Sintaxis

Delphi `myReal := Objeto.Temperatura[0].ValorPreMaximo;`  
Visual Basic `myFloat = Objeto.Temperatura(0).ValorPreMaximo`  
(El índice 0 indica el primer canal de medición de temperaturas)

#### Modo

Lectura/escritura.

**Tipo de dato**

Delphi	Real
Visual Basic	Float

**Valores Posibles**

MIN\_TEMP\_VALUE .. MAX\_TEMP\_VALUE (ver constantes)

**Propiedad ValorMinimo**

Si esta propiedad tiene un valor no nulo, el RmtX generará un evento **OnAlarmaTemp** cuando la medición esté por debajo de su valor.

**Sintaxis**

Delphi	myReal := Objeto. <b>Temperatura[0].ValorMinimo</b> ;
Visual Basic	myFloat = Objeto. <b>Temperatura(0).ValorMinimo</b>

(El índice 0 indica el primer canal de medición de temperaturas)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi	Real
Visual Basic	Float

**Valores Posibles**

MIN\_TEMP\_VALUE .. MAX\_TEMP\_VALUE (ver constantes)

**Propiedad ValorPreMinimo**

Si esta propiedad tiene un valor no nulo, el RmtX generará un evento **OnPreAlarmaTemp** cuando la medición esté por debajo de su valor, y por encima del ValorMinimo.

**Sintaxis**

Delphi	myReal := Objeto. <b>Temperatura[0].ValorPreMinimo</b> ;
Visual Basic	myFloat = Objeto. <b>Temperatura(0).ValorPreMinimo</b>

(El índice 0 indica el primer canal de medición de temperaturas)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi	Real
Visual Basic	Float

**Valores Posibles**

MIN\_TEMP\_VALUE .. MAX\_TEMP\_VALUE (ver constantes)

**Propiedad MantieneRegistro**

El WebPort Remote posee la capacidad de almacenar las últimas MAX\_REGS\_INP mediciones de temperatura para cada uno de sus canales de sensado.

Esta propiedad determina si se registrarán los cambios de temperatura del canal indicado, para su posterior acceso mediante la propiedad **Valores**.

**Sintaxis**

Delphi                    myBoolean := Objeto.**Temperatura[0].MantieneRegistro**;  
 Visual Basic            myBoolean = Objeto.**Temperatura(0).MantieneRegistro**  
 (El índice 0 indica el primer canal de medición de temperaturas)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi                    Boolean  
 Visual Basic            Boolean

**Valores Posibles**

True:                    Mantiene registro.  
 False:                  No mantiene registro.

**Propiedad IntervaloMuestreo**

Esta propiedad determina el intervalo de interrogación al WebPort Remote para tomar mediciones de temperatura.

Si bien el Remote mantiene el estado cuasi-instantáneo de las temperaturas, la aplicación puede ajustar el tiempo de actualización de esta información, lo que determinará la generación de los eventos relacionados a la temperatura.

Las unidades de este intervalo son segundos.

Si este valor está en cero, no se generarán evento de mediciones de temperaturas.

**Sintaxis**

Delphi                    myInteger := Objeto.**Temperatura[0].IntervaloMuestreo**;  
                              Objeto.**Temperatura[0].IntervaloMuestreo** := myInteger;  
 Visual Basic            myLong = Objeto.**Temperatura(0).IntervaloMuestreo**  
                              Objeto.**Temperatura(0).IntervaloMuestreo** = myLong  
 (El índice 0 indica el primer canal de medición de temperaturas)

**Modo**

Lectura/escritura.

**Tipo de dato**

Delphi                    Integer  
 Visual Basic            Long

**Valores Posibles**

0 .. 2147483647  
 Valor por defecto:    **5 segundos.**

**Propiedad Valores**

Si la propiedad **MantieneRegistro** está en True, el WebPort Remote acumulará internamente hasta MAX\_REGS\_TEMP mediciones de temperatura para cada uno de los 4 canales disponibles.

Los valores registrados, junto con la fecha y hora del registro se acceden mediante 2 propiedades:

**Propiedad Temperatura****Sintaxis**

Delphi                    myReal := Objeto.**Temperatura[0].Valores[n].Temperatura**;  
 Visual Basic            myFloat = Objeto.**Temperatura(0).Valores(n).Temperatura**  
 (El índice 0 indica el primer canal de medición de temperaturas)  
 (El índice n indica la enésima medición registrada)

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Real
Visual Basic	Float

**Valores Posibles**

MIN\_TEMP\_VALUE .. MAX\_TEMP\_VALUE (ver constantes)

**Propiedad FechaHora****Sintaxis**

Delphi	myDateTime := Objeto. <b>Temperatura[0].Valores[n].FechaHora</b> ;
Visual Basic	myFloat = Objeto. <b>Temperatura(0).Valores(n).FechaHora</b> (El índice 0 indica el primer canal de medición de temperaturas) (El índice <b>n</b> indica la enésima medición registrada)

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	tDateTime
Visual Basic	Date

**Evento OnAlarmaTemp**

Cuando uno de los canales de medición de temperatura está **Habilitado**, y supera el valor definido en **ValorMaximo** o está por debajo del valor definido en **ValorMinimo**, se genera un evento **OnAlarmaTemp**.

Este evento pasa como parámetro el número de canal en el que se produjo la alarma.

**Sintaxis**

Delphi	Sub Objeto. <b>OnAlarmaTemp</b> (Canal: integer);
Visual Basic	Sub Objeto. <b>OnAlarmaTemp</b> (ByVal Canal: Long)

**Argumento**

Delphi 4	Canal: Integer, indica el canal en alarma.
Visual Basic	Canal: Long, indica el canal en alarma.

**Evento OnPreAlarmaTemp**

Cuando uno de los canales de medición de temperatura está **Habilitado**, y el valor medido se ubica en la franja ubicada entre **ValorPreMaximo** y **ValorMaximo** o en la banda ubicada entre **ValorPreMinimo** y **ValorMinimo**, se genera un evento **OnPreAlarmaTemp**.

Este evento pasa como parámetro el número de canal en el que se produjo la alarma.

**Sintaxis**

Delphi	Sub Objeto. <b>OnPreAlarmaTemp</b> (Canal: integer);
Visual Basic	Sub Objeto. <b>OnPreAlarmaTemp</b> (ByVal Canal: Long)

**Argumento**

Delphi 4	Canal: Integer, indica el canal en alarma.
Visual Basic	Canal: Long, indica el canal en alarma.

## Evento OnRetornoNormalTemp

Cuando uno de los canales de medición de temperatura está **Habilitado**, y retorna a un valor normal, se genera un evento **OnRetornoNormalTemp**.

Este evento pasa como parámetro el número de canal que retorna a la normalidad.

### Sintaxis

Delphi	Sub Objeto. <b>OnRetornoNormalTemp</b> (Canal: integer);
Visual Basic	Sub Objeto. <b>OnRetornoNormalTemp</b> (ByVal Canal: Long)

### Argumento

Delphi 4	Canal: Integer, indica el canal recuperado de una alarma.
Visual Basic	Canal: Long, indica el canal recuperado de una alarma.

## 1.6.7 Registros digitales

El WebPort Remote mantiene el registro de hasta 1024 estados de los canales digitales. Puede determinarse el modo de actualización de estos registros (si se actualizan cíclicamente o se mantienen hasta ser borrados por la aplicación).

El Remote insertará un nuevo registro cada vez que se produzca una variación en cualquiera de las entradas digitales activas.

A su vez, el registro contiene el estado conjunto de los 16 canales, independientemente de cuales sean los que se encuentran bajo control.

El manejo de los registros de estados digitales se implementa mediante la propiedad **Registros**

### Propiedad Habilitado

Esta propiedad refleja si se procesan los registros y sus eventos asociados.

### Sintaxis

Delphi	Objeto. <b>Registros.Habilitado</b> := myBoolean; myBoolean := Objeto. <b>Registros.Habilitado</b> ;
Visual Basic	Objeto. <b>Registros.Habilitado</b> = myBoolean myBoolean = Objeto. <b>Registros.Habilitado</b>

### Modo

Lectura/escritura.

### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

### Valores Posibles

True:	Registros digitales habilitados.
False:	Registros digitales deshabilitados.

### Propiedad Ciclico

Esta propiedad determina el comportamiento de los registros, cuando se llega al límite de 1024 datos acumulados.

Si el valor asignado a **Ciclico** es True, los datos siguientes se irán acumulando sobre los registros mas viejos, actualizandose en forma cíclica.

Si, por el contrario, este valor está en False, los cambios digitales subsiguientes no se registran, y se lleva la cuenta de registros perdidos en la propiedad **RegistrosPerdidos**.

### Sintaxis

Delphi	Objeto. <b>Registros.Ciclico</b> := myBoolean; myBoolean := Objeto. <b>Registros.Ciclico</b> ;
Visual Basic	Objeto. <b>Registros.Ciclico</b> = myBoolean myBoolean = Objeto. <b>Registros.Ciclico</b>

### Modo

Lectura/escritura.

### Tipo de dato

Delphi	Boolean
Visual Basic	Boolean

### Valores Posibles

True:	Registros nuevos sobrescriben a los viejos.
False:	Registros nuevos se pierden.

## Propiedad Perdidos

Indica la cantidad de registros perdidos desde la última inicialización.

Si se ha indicado la propiedad Ciclico en False, los cambios en los estados digitales que se producen una vez acumulados los 1024 registros posibles se descartan. En ese caso, la cantidad descartada se acumula en este contador.

El RmtX pondrá en cero este valor cuando se lleve a cabo una lectura.

### Sintaxis

Delphi	myInteger := Objeto. <b>Registros.Perdidos</b> ;
Visual Basic	myLong = Objeto. <b>Registros.Perdidos</b>

### Modo

Solo lectura.

### Tipo de dato

Delphi	Integer
Visual Basic	Long

### Valores Posibles

0 .. 2147483647

## Propiedad UmbralAlarma

Indica el número de registro (0..1023) en el que se genera un evento para avisar que se ha llegado a un punto determinado de la acumulación.

El evento a ser generado es **OnMaxRegistros**.

### Sintaxis

Delphi	mySmallint := Objeto. <b>Registros.UmbralAlarma</b> ;
Visual Basic	myInteger = Objeto. <b>Registros.UmbralAlarma</b>

### Modo

Solo lectura.

**Tipo de dato**

Delphi	Smallinteger
Visual Basic	Integer

**Valores Posibles**

0.. 1023                      Número de registro que determina el evento.

**Propiedad Cantidad**

Indica la cantidad de registros acumulados.

Una vez llegado al registro 1023 , este dato permanece en 1023 hasta que se invoque el método

**BorrarRegistros.**

**Sintaxis**

Delphi	myInteger := Objeto. <b>Registros.Cantidad</b> ;
Visual Basic	myLong = Objeto. <b>Registros.Cantidad</b>

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Integer
Visual Basic	Long

**Valores Posibles**

0..1023:                      Cantidad de registros acumulados.

**Objeto Datos**

Si la propiedad **MantieneRegistro** está en True, el WebPort Remote acumulará internamente hasta 1024 registros de los estados digitales.

Los estados registrados, junto con el instante del registro se acceden mediante 3 propiedades:

**Propiedad Estado**

Esta propiedad representa el estado conjunto de los 16 canales posibles.

El bit 0 corresponde al estado del canal digital 0, y así hasta el bit 15.

Para acceder al estado de un canal en particular, puede utilizarse el método **BitRegistro.**

**Sintaxis**

Delphi	myWord := Objeto. <b>Registros.Datos[n].Estado</b> ;
Visual Basic	myInteger = Objeto. <b>Registros.Datos(n).Estado</b> (El índice <b>n</b> indica el enésimo registro)

**Modo**

Solo lectura.

**Tipo de dato**

Delphi	Word
Visual Basic	Integer

**Propiedad Segundos**

Indica los segundos del evento, contados desde la puesta en marcha del webport.

**Sintaxis**

Delphi	myInteger := Objeto. <b>Registros.Datos[n].Segundos</b> ;
--------	---

Visual Basic                      myLong = Objeto.**Registros.Datos(n).Segundos**  
(El índice **n** indica el enésimo registro)

**Modo**

Solo lectura.

**Tipo de dato**

Delphi                                      Integer  
Visual Basic                                Long

**Propiedad Msegs**

Indica los milisegundos que hay que adicionarse al segundo referido en la propiedad anterior, para obtener el evento con mayor precisión.

**Sintaxis**

Delphi                                      myInteger := Objeto.**Registros.Datos[n].Msegs**;  
Visual Basic                                myLong = Objeto.**Registros.Datos(n).Msegs**  
(El índice **n** indica el enésimo registro)

**Modo**

Solo lectura.

**Tipo de dato**

Delphi                                      Integer  
Visual Basic                                Long

**Método Borrar**

Borra los primeros n elementos de la secuencia de registros.  
El parámetro puede ir de 0 a 1024.

**Sintaxis**

Delphi                                      Function Objeto.**BorrarRegistros**(Cantidad: Integer): Integer;  
Visual Basic                                Function Objeto.**BorrarRegistros**(Cantidad as Long) As Long

**Argumento**

La cantidad de registros a borrar, comenzando desde el primer registro (índice cero).

Delphi                                      Cantidad: Integer (0 .. 1024)  
Visual Basic                                Cantidad: Long (0 .. 1024)

**Resultado**

Se obtiene un código de estado:

0:    éxito.  
-1:     comando inválido.  
-2:     timeout.  
-3:     dirección inválida.  
-4:     error de comunicaciones.

**Método BitRegistro**

Accede al estado de un bit particular de un registro de estados digitales.

**Sintaxis**

Delphi                                      Function Objeto.**BitRegistro**(Registro: Integer ; bit: Integer): Boolean;

Visual Basic                      Function Objeto.**BitRegistro**(Registro As Long , bit As Long) As Boolean;

### Argumentos

La posición del registro a leer y el número de bit .  
Delphi                              Registro: Integer. [0..1023]  
   Bit: Integer. [0..15]  
Visual Basic                      Cantidad: Long. [0..1023]  
   Bit: Long. [0..15]

### Resultado

True:                                Bit On  
False:                               Bit Off.

### Evento OnMaxRegistros

Al superar la cantidad de registros determinados mediante la propiedad **RegistroAlarma**, se generará este evento.

### Sintaxis

Delphi                                Sub Objeto.**OnMaxRegistros**();  
Visual Basic                        Sub Objeto.**OnMaxRegistros**()

### Argumento

No posee

## 1.7 Resumen

### 1.7.1 Propiedades

#### Generales

Activo  
IP  
TiempoDeMarcha  
Contacto  
Identificacion  
Ubicacion  
ModoES  
Entradas  
FechaHora  
VersionRmtX  
VersionFirmware  
MAX\_ENTRADAS\_DIG  
MAX\_COM\_PORTS  
MAX\_SALIDAS\_DIG  
MAX\_SENSORES\_TEMP

#### Comunicaciones

ComPorts(0..2).Activo  
ComPorts(0..2).Velocidad  
ComPorts(0..2).Paridad  
ComPorts(0..2).Bits  
ComPorts(0..2).StopBits

#### Entradas Digitales

Entradas(0..15).Descripcion  
Entradas(0..15).DescripcionAlarma  
Entradas(0..15).Habilitada  
Entradas(0..15).EstadoNormal  
Entradas(0..15).EstadoActual  
Entradas(0..15).Acumulador  
Entradas(0..15).Contador  
Entradas(0..15).Maximo  
Entradas(0..15).PreMaximo  
Entradas(0..15).GeneraAlarmas  
Entradas(0..15).ResetEnMaximo  
Entradas(0..15).IntervaloMuestreo

#### Salidas Digitales

Salidas(0..7).Descripcion  
Salidas(0..7).EstadoBase  
Salidas(0..7).EstadoActual

#### Temperaturas

Temperaturas[0..3].Habilitado  
Temperaturas[0..3].Descripcion  
Temperaturas[0..3].DescripcionAlarma  
Temperaturas[0..3].Conectado  
Temperaturas[0..3].IdTermometro  
Temperaturas[0..3].ValorActual  
Temperaturas[0..3].ValorMaximo  
Temperaturas[0..3].ValorPreMaximo

Temperaturas[0..3].ValorMinimo  
Temperaturas[0..3].ValorPreMinimo  
Temperaturas[0..3].MantieneRegistro  
Temperaturas[0..3].IntervaloMuestreo  
Temperaturas[0..3].Valores[0..MAX\_REGS\_TEMP - 1].Temperatura  
Temperaturas[0..3].Valores[0..MAX\_REGS\_TEMP - 1].FechaHora

### **Registros Digitales**

Registros.Habilitado  
Registros.Ciclico  
Registros.Perdidos  
Registros.UmbralAlarma  
Registros.Cantidad  
Registros.Datos[0..MAX\_REGS\_INP - 1].Estado  
Registros.Datos[0..MAX\_REGS\_INP - 1].FechaHora

## **1.7.2 Eventos**

### **Generales**

OnError

### **Comunicaciones**

OnDataIn

### **Entradas Digitales**

OnAlarmaDigital  
OnContadorMaximoAlcanzado  
OnContadorPreMaximoAlcanzado  
OnRetornoDigitalNormal

### **Temperaturas**

OnAlarmaTemp  
OnPreAlarmaTemp  
OnRetornoNormalTemp

### **Registros Digitales**

OnMaxRegistros

### 1.7.3 Métodos

#### Comunicaciones

PutString  
PutChar  
GetChar  
CharReady

#### Registros Digitales

Borrar  
BitRegistro

## 2 Ejemplo en Visual Basic 6.0

### 2.1 Instalación y funcionamiento

Con el CD de instalación, se provee un ejemplo simple de aplicación del componente RmtX.

**los archivos del mismo son** PruebaRmt.frm, PruebaRmt.vbp y PruebaRmt.vbw

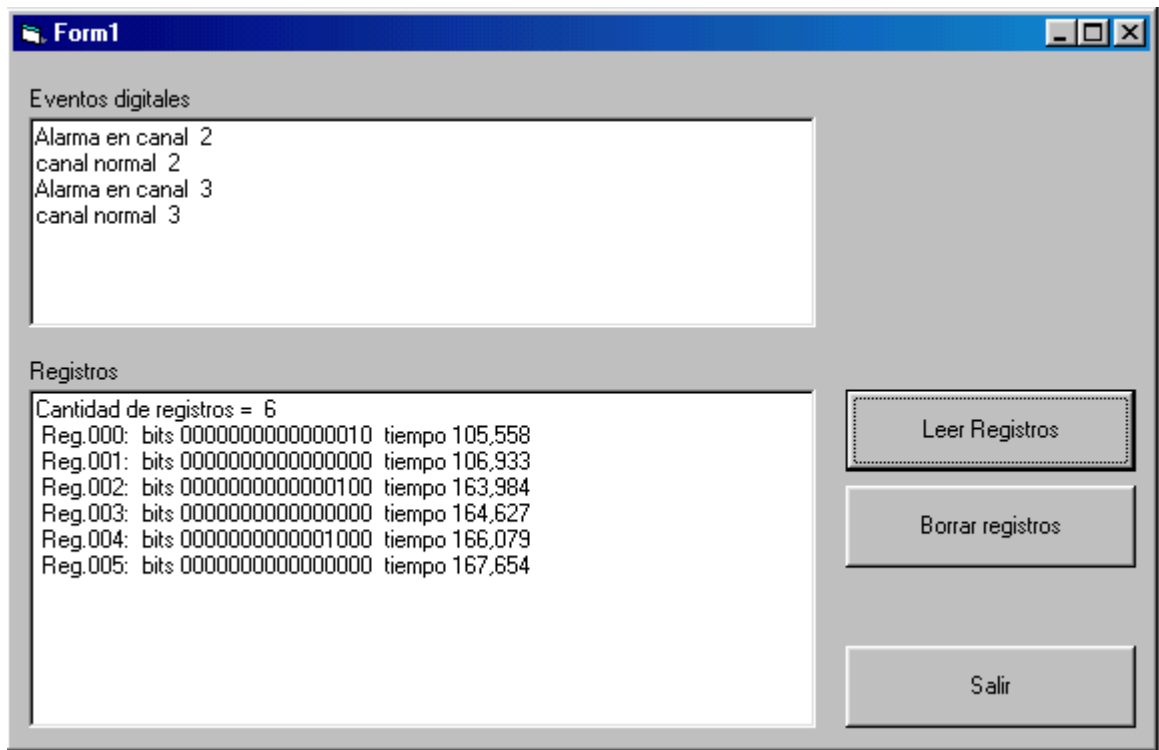
Cree un directorio específico para este proyecto, y copie los 3 archivos al mismo.

Una vez instalado el componente RmtX, puede abrir el proyecto PruebaRmt y ejecutarlo. (Para su funcionamiento real, requiere que el WebPort remote esté conectado, y que la PC donde se testea el proyecto tenga conectividad TCP/IP con el Remote).

Si el WebPort no está encendido, o si no hay comunicación con el IP respectivo, al ejecutar el proyecto se recibirá el mensaje:



Una vez chequeadas las comunicaciones, se presenta la ventana del ejemplo:



El ejemplo muestra en el editor superior los eventos o las alarmas, a medida que las mismas se producen.

Presionando el botón Leer Registros, se visualizan los registros de los eventos acumulados (hasta 1024), con el instante de su detección.

El tiempo de todos los registros es relativo al encendido del WebPort remote, y se muestra combinando los segundos y los milisegundos.

En el ejemplo se utiliza una función (`Function LongtoBin(Valor As Long) As String`) para mostrar el estado de los 16 bits en una sola línea (el 1er bit corresponde al canal 15, y el último al canal 0).

La interpretación de los registros es:

registro 002 : a los 163 s 984 ms se activó el canal 2

registro 003 : a los 164 s 627 ms el canal 2 retornó al estado normal

a esos 2 registros corresponden los 2 primeros mensajes de la ventana de eventos

registro 004 : a los 166 s 079 ms se activó el canal 3

registro 005 : a los 167 s 654 ms se activó el canal 3

El botón Borrar registros lee la cantidad actual de registros y luego manda a borrar dicha cantidad (borra todos los registros).

Finalmente, el botón Salir, finaliza la aplicación.

## 2.2 Código fuente

```

Function LongtoBin(Valor As Long) As String
    Dim bit As Integer
    Dim Pot2 As Long
    Dim resultado As String

    resultado = ""
    Pot2 = 1
    For bit = 1 To 16
        If (Valor And Pot2) = 0 Then
            resultado = "0" + resultado
        Else
            resultado = "1" + resultado
        End If
        Pot2 = 2 * Pot2
    Next
    LongtoBin = resultado
End Function

Function FormatRegistro(Datos As RegistroDigital) As String
    FormatRegistro = " bits " + LongtoBin(Datos.Estado) + " tiempo " +
    Format((Datos.Segundos + Datos.Msecs / 1000), "#####0.000")
End Function

Private Sub Command1_Click()
    Dim Registro As RegistroDigital
    CantidadRegistros = Remote.Registros.Cantidad
    txtRegistros = "Cantidad de registros = " + Str$(CantidadRegistros) + vbCrLf
    For j = 0 To CantidadRegistros - 1
        Registro = Remote.Registros.Datos(j)
        txtRegistros = txtRegistros + " Reg." + Format(j, "000") + ": " +
        FormatRegistro(Registro) + vbCrLf
    DoEvents
    Next
End Sub

Private Sub Command2_Click()
    End
End Sub

Private Sub Command3_Click()
    Remote.Registros.Borrar (Remote.Registros.Cantidad)
End Sub

Private Sub Form_Load()
    'seteo en el que esta ubicado el remote
    'este valor hay que modificarlo en caso de cambiar el IP del equipo
    Remote.IP = "10.110.110.100"

    'en el momento de cargar el form pongo todas las entradas digitales
    'en generar alarams (para que los cambios de cualquier entrada digital
    'disparen los eventos OnAlarmaDigital y OnRetornoDigitalNormal)

    'Asimismo pido que se verifique el estado de las entradas digitales cada
    '3 decimas de segundo.
    'El intervalo de muestreo es el tiempo entre interrogaciones al remote desde
    'este programa.
    'Independientemente de cuanto diga el intervalo de muestreo el remote guarda
    'todas las transiciones mirando sus entradas digitales cada 0.7 mS

    For j = 0 To Remote.MAX_ENTRADAS_DIG
        Remote.Entrada(j).IntervaloMuestreo = 5
        Remote.Entrada(j).GeneraAlarmas = True
    Next

```

```
'al poner la propiedad activo en true se lleva a cabo la comunicacion con
'el equipo
Remote.Activo = True
End Sub

Private Sub Remote_OnAlarmaDigital(ByVal Canal As Integer)
'agrego una linea al text box de eventos cada vez que cambia un canal
txtEventos.Text = txtEventos.Text & "Alarma en canal " & Str$(Canal) & vbCrLf
End Sub

Private Sub Remote_OnRetornoDigitalNormal(ByVal Canal As Integer)
'agrego una linea al text box de eventos cada vez que cambia un canal
txtEventos.Text = txtEventos.Text & "canal normal " & Str$(Canal) & vbCrLf
End Sub
```

## 2.3 Comentarios

### Comentarios sobre las rutinas del programa de ejemplo

**Function LongtoBin(Valor As Long) As String**

Transforma el resultado de una lectura de estados digitales (que es una variable de tipo Longh), en un string de 16 bytes, expresando en formato de ceros o unos el estado conjunto de las entradas.

**Function FormatRegistro(Datos As RegistroDigital) As String**

Prepara los datos de un registro para su visualización:

- Pasa el estado a formato de ceros y unos.
- Toma los segundos y los milisegundos y los combina en una única expresión de tiempo.

**Private Sub Command1\_Click()** (Botón Leer registros)

Lleva a cabo la lectura de todos los registros acumulados en el WebPort.

- Interroga al Remote y obtiene el número de registros acumulados.
- En un loop desde el registro inicial hasta la cantidad menos 1, toma el registro enésimo.
- El registro (asociado a una variable de tiempo RegistroDigital) se formatea mediante las funciones ya descritas, y se muestra.

El comando DoEvents, permite actualizar los componentes visuales a medida que se llevan a cabo las lecturas.

**Private Sub Command2\_Click()** (Botón Salir)

Finaliza la aplicación.

**Private Sub Command3\_Click()** (Botón Borrar registros)

Manda a borrar el número actual de registros (borra todos los registros).

**Private Sub Form\_Load()**

Al inicializarse el form, se llevan a cabo las inicializaciones necesarias.

Este es el punto del proyecto en el que se debe configurar el comportamiento deseado del WebPort. (Ver los comentarios del código fuente).

**Private Sub Remote\_OnAlarmaDigital(ByVal Canal As Integer)**

Este evento se produce cada vez que el WebPort genera una alarma, esto es, cada vez que algún canal digital cambia de su estado normal al estado complementario.

En el proyecto, esto genera una línea en la ventana de eventos.

**Private Sub Remote\_OnRetornoDigitalNormal(ByVal Canal As Integer)**

Este evento se produce cada vez que el WebPort retorna de una alarma, esto es, cada vez que algún canal digital recupera su estado normal,

En el proyecto, esto genera una línea en la ventana de eventos.