TRADING TOOLS

MANUALES PARA EL USUARIO

visualchart

CONTENIDO

Introducción

¿Que son las Trading Tools?

Las Trading Tools de Visual Chart V son unas potentes herramientas basadas en la tecnología COM (Component Object Model), que permiten acceder a la información que se maneja desde el programa, a través de cualquier entorno de desarrollo compatible con dicha tecnología.

La herramienta COM actúa como intermediaria entre Visual Chart y la aplicación cliente, por ejemplo Microsoft Excel.

Esta imagen resume cómo se produce la comunicación entre Visual Chart y la aplicación cliente, de manera que el programa que actúa como cliente realiza peticiones a la herramienta COM y ésta notifica los resultados a través de eventos.

Las distintas librerías que forman las **Trading Tools** contienen diversos objetos que heredan las diferentes modalidades de tratamiento de datos que se pueden usar desde Visual Chart, tales como:

- Acceder a la información en tiempo real de símbolos
- Acceder a la fuente histórica de datos de cualquier símbolo
- Obtener datos técnicos tales como volumen, indicadores o posiciones de sistemas
- Acceder a la información procedente de las contribuciones
- Realizar operaciones reales a mercado

Al interactuar con los servidores de datos e intermediación, **es necesario que Visual Chart esté conectado Realserver y al servidor de intermediación (operativa simulada o real).**

Para comprender el funcionamiento de estas herramientas, es necesario entender los siguientes conceptos:

OBJETO Es un conjunto de información y funcionalidades relacionados entre sí. Representa un concepto, y contiene toda la información necesaria para abstraerlo:

- Datos que describen sus atributos
- Operaciones que pueden realizarse sobre los mismos

EVENTO es la forma de comunicarse que tiene un objeto.

MÉTODO es una función que permite actuar sobre un objeto.

En este manual utilizaremos Microsoft Excel para realizar algunos ejemplos que expliquen el uso de las Trading Tools.

No obstante, como se ha indicado anteriormente, se puede utilizar con otras aplicaciones clientes compatibles con la tecnología COM.

Configuración previa de Excel antes de la programación

Será necesario crear un libro Excel nuevo donde poder implementar nuestras herramientas. Cabe destacar que puesto que vamos a desarrollar **macros en Excel** que utilicen los servidores COM, será necesario que la aplicación Excel tenga activada la habilitación de macros, además de tener desactivado el control de usuarios (Windows Vista o versiones superiores).

EVITAR UN BLOQUEO POR CONTROL DE USUARIOS

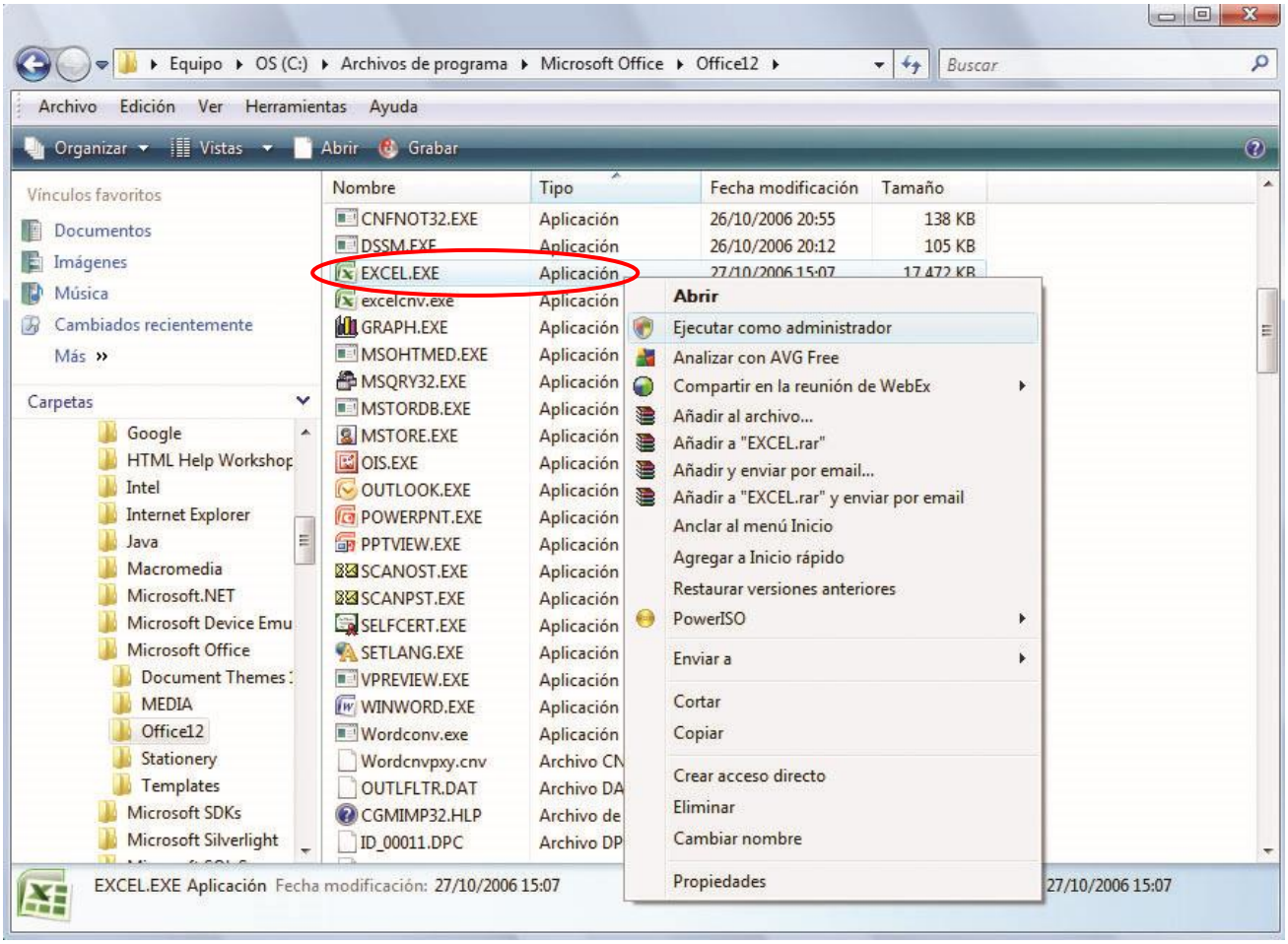
Para evitar posibles bloqueos por el control de usuarios, es aconsejable abrir Excel como administrador. A continuación se indican los pasos a seguir:

1º Acceder a Excel a través de la ruta:

C:\Archivos de programa\Microsoft Office\Office12

Es posible que en lugar de la carpeta **Office12** exista una carpeta llamada **Office11.** En tal caso los pasos a realizar serán los mismos.

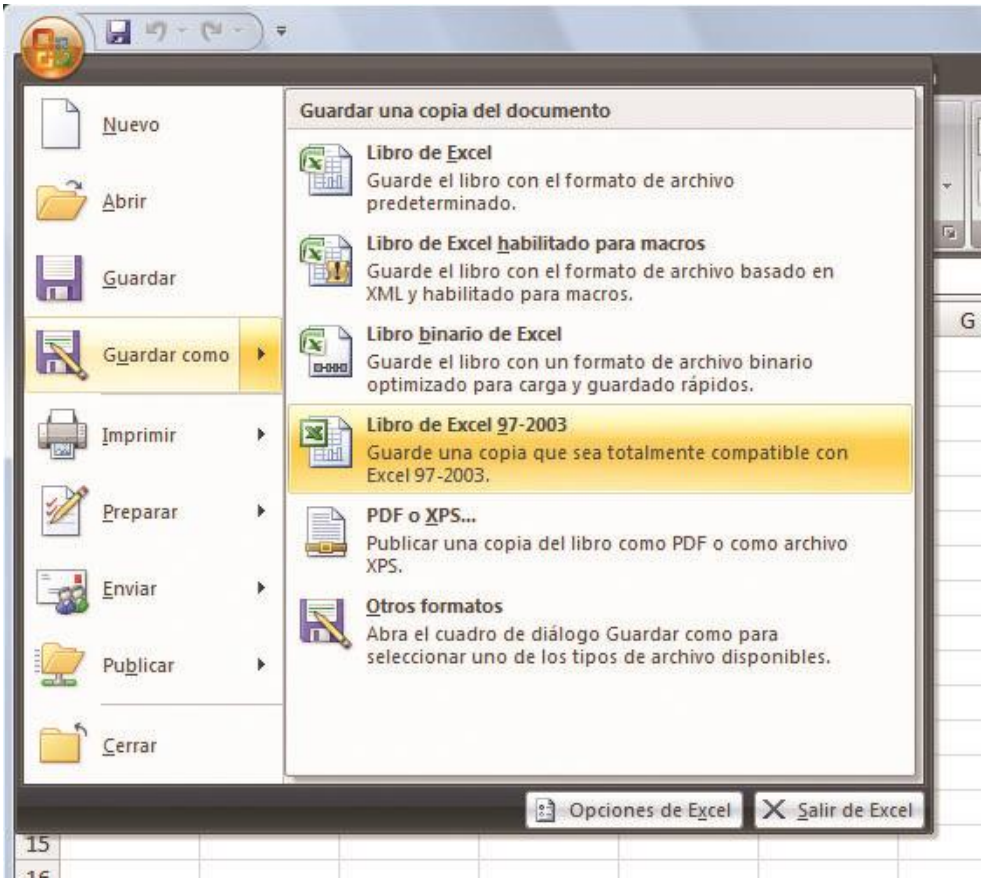
2º Una vez dentro de la carpeta, es necesario buscar el icono denominado **EXCEL** de tipo aplicación y ejecutarlo como administrador. Para esto tan sólo hay que pulsar sobre el icono con el botón derecho del ratón, y hacer clic sobre la opción **Ejecutar como administrador** del menú contextual.



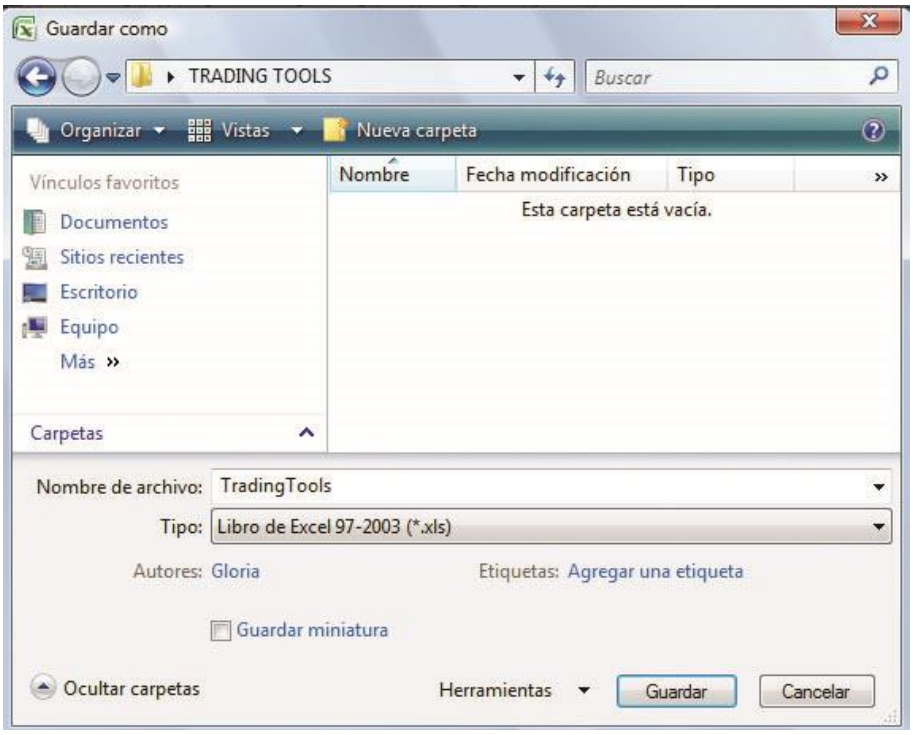
Hecho esto tendremos privilegios de Windows para usar la aplicación.

EVITAR CONFLICTOS CON VERSIONES ANTERIORES DE OFFICE

Es recomendable abrir un libro nuevo y guardarlo como **Libro de Excel 97-2003**. De esta forma eliminamos posible problemas de incompatibilidad con otras versiones anteriores de Office.



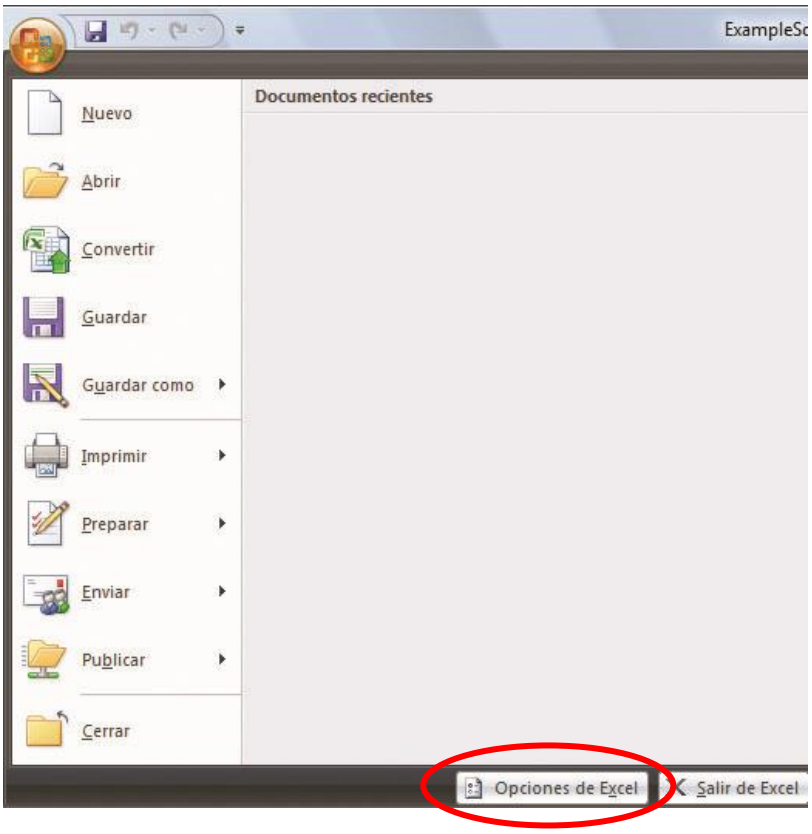
Para el ejemplo lo guardaremos con el nombre **TradingTools.xls**



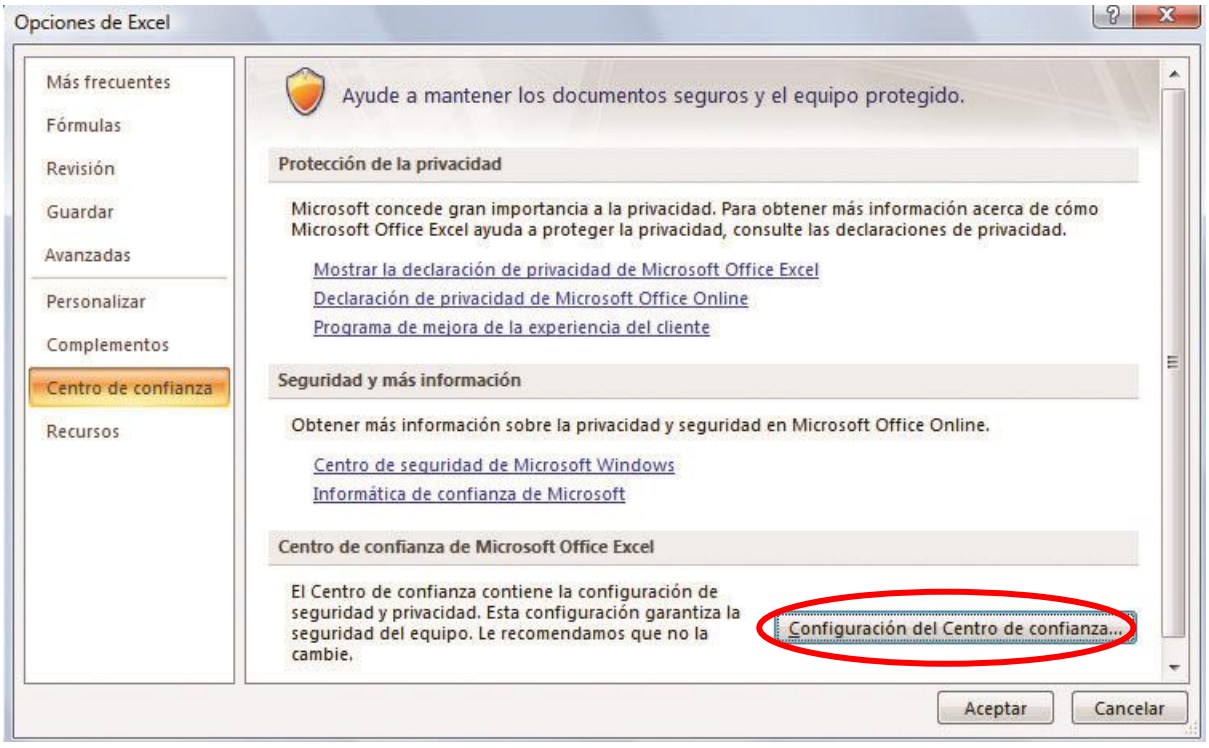
HABILITAR DE MACROS

Para habilitar macros es preciso seguir los pasos indicados a continuación:

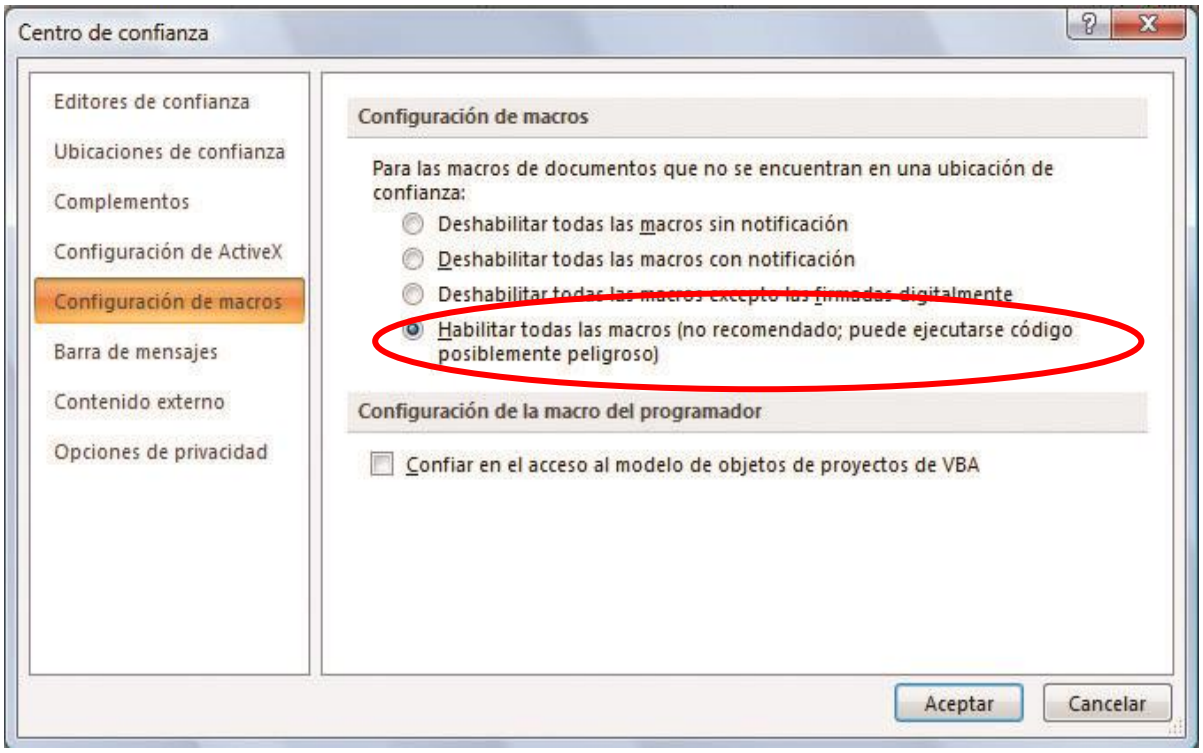
1º Acceder a las **opciones de Excel** a través desde el menú principal



2º En el cuadro de diálogo es necesario seleccionar el apartado **Centro de confianza** y pulsar el botón **Configuración del Centro de confianza...**



3º Seleccionar la opción **Configuración de macros** y activar la casilla **Habilitar todas las macros**



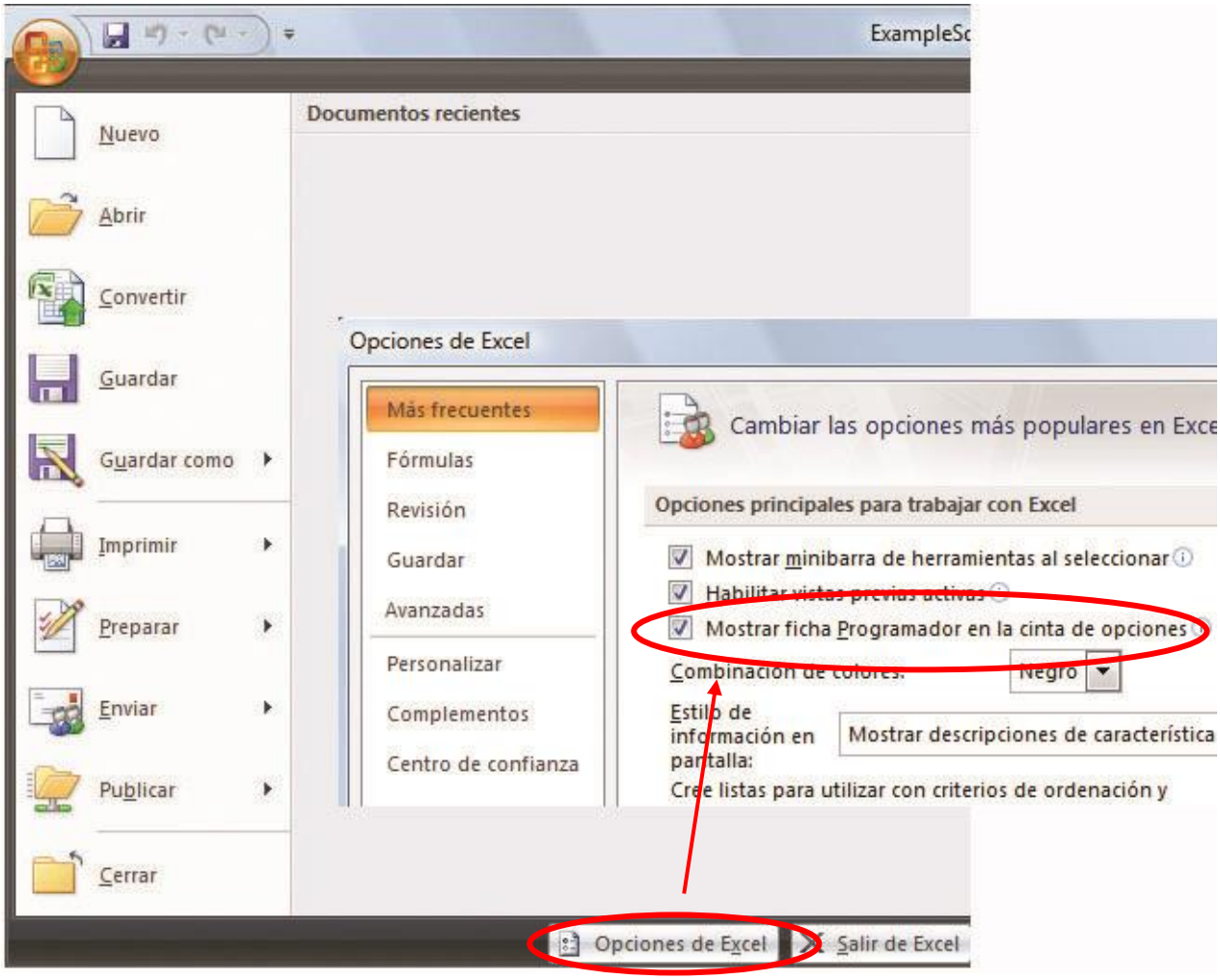
4º Para finalizar pulsar sobre el botón **Aceptar.**

AÑADIR LIBRERÍAS DE VISUAL CHART

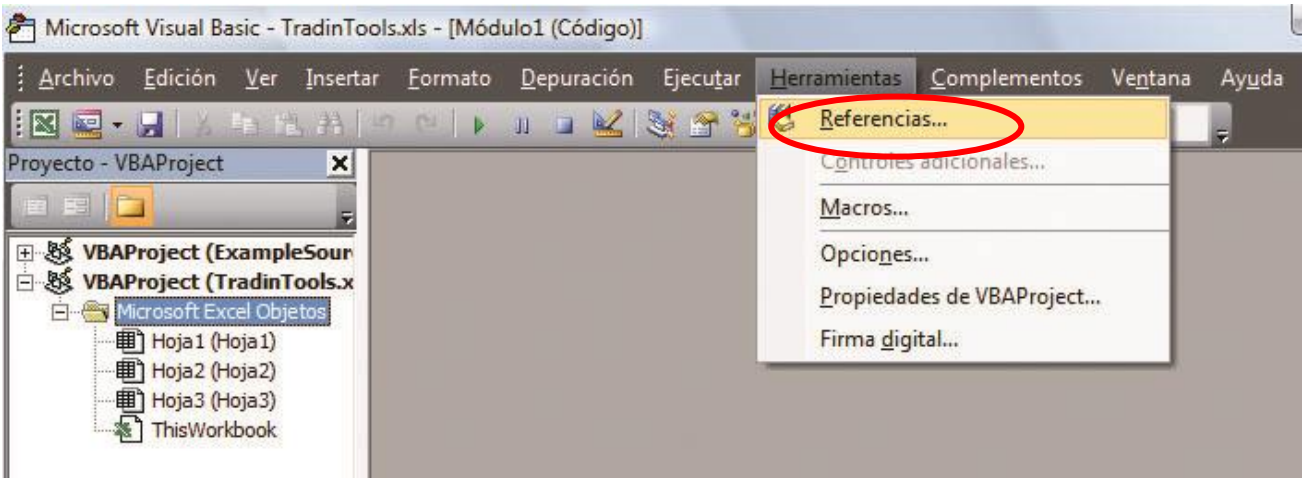
Para la utilización de las Trading Tools en Excel, es necesario registrar las librerías que se van a utilizar. Esto lo haremos de la siguiente forma:

1º Accionar sobre el comando de Visual Basic del menú **Programador**.

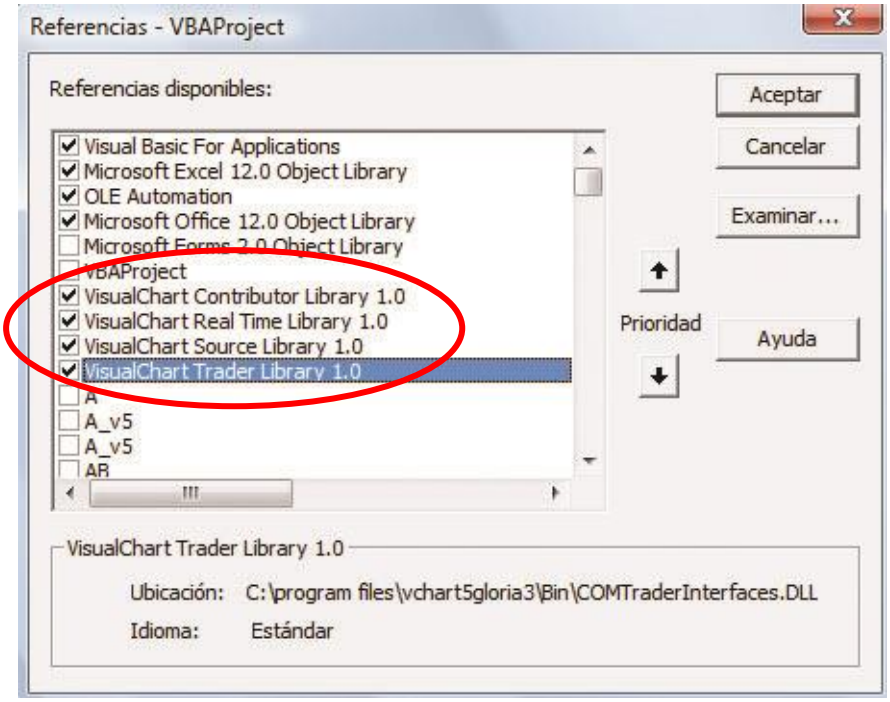
Si en la cinta de opciones no se mostrara el menú **Programador**, el usuario puede incorporarlo a través de las **Opciones de Excel** del menú principal. Es necesario activar la opción **Mostrar ficha programador en la cinta de opciones,** la cual se encuentra en la categoría **Más Frecuentes.**



2º Seleccionar en el menú **Herramientas** del editor de Visual Basic la opción **Referencias...** (es posible que tarde unos segundos la carga de las librerías).



En la lista de referencias disponibles será necesario buscar las 4 librerías COM y activarlas. De esta forma se podrán utilizar los diferentes eventos, objetos y métodos.



Para finalizar pulsaremos el botón **Aceptar**.

A partir de este momento, las librerías quedan cargadas en Excel y listas para poder ser utilizarlas. En caso de usar cual cualquier otra aplicación, igualmente debemos buscar la opción de referencias para poder activar las librerías COM.

3º Abrir Visual Chart V (en caso de estar cerrado) y asegurarse de que se dispone de información. Hay que tener en cuenta que el usuario sólo dispondrá en la aplicación cliente de aquella información a la que tenga acceso en su licencia de Visual Chart.

SERVIDORES COM

VCRealTime

INTRODUCCIÓN

VCRealTime proporciona acceso a los datos en tiempo real de los diferentes mercados a los que se tenga acceso desde Visual Chart. Para su funcionamiento, es necesario que el programa se esté ejecutando y además esté conectado al servidor de datos (RealServer).

Existen dos métodos a través de los cuales el servidor **VCRealTime** facilita la información al cliente:

A. Notificación mediante la recepción de eventos. Cada vez que se genera un nuevo valor, el cliente recibe un evento con la información resultante.

B. Solicitud del último valor que guarde el servidor por parte del cliente mediante métodos.

En ambos casos, el cliente previamente ha debido realizar peticiones a **VCRealTime** de aquellos valores de los que desea recibir datos en tiempo real.

El cliente puede solicitar información de diversa índole:

- Campos de tiempo real
- Posiciones de compra/venta
- Datos fundamentales calculados
- Análisis técnico
- Información del símbolo
- Etc.

En definitiva, el servidor **VCRealTime** proporciona cualquier dato de los que se pueden tener acceso a través de las tablas de Visual Chart.

Para poder hacer uso de este servidor, la aplicación cliente debe de tener incorporada a su lista de referencias la clase **VisualChart Real Time Library.dll**

Esta librería incluye una serie de objetos, eventos y métodos que a continuación se detallan.

OBJETOS

Los objetos de la librería **VCRealTime** son los siguientes:

- VCRT_ Tick
- VCRT_Limit
- VCRT_RealTime

VCRT_Tick. Contienen la información del campo que ha cambiado para un símbolo determinado.

Propiedad	Descripción
SymbolCode As String	Valor (código del valor) asociado.
Date As Date	Fecha del cambio.
Field As enumVCRTField	Campo que ha cambiado.
Text As String	Valor que tiene actualmente el campo.
Value As Double	Valor que tiene actualmente el campo.
TickIndex As Long	Índice del tick 1.
FieldEx As Long	Campo extendido 2.

Para recibir esta información es necesario invocar previamente alguno de estos métodos:

- **RequestSymbolFeed/RequestSymbolsFeed** para los campos de enumVCRTField
- **RequestFieldEx/RequestFieldsEx** para los campos extendidos.

Una vez recibidos, se pueden consultar estos valores en cualquier momento mediante los métodos **GetFieldText/GetFieldValue** o **GetFieldExText/ GetFieldExValue.**

Por otro lado, el evento **OnNewTicks** devuelve un objeto VCRT_Tick conteniendo los últimos valores reportados.

1. Los ticks que tienen la misma fecha (se han producido en el mismo segundo), se distinguen por un valor autonumérico que se refleja en el campo TickIndex.

2. Un campo extendido engloba un número considerable de posibles valores y puede ir creciendo en cualquier momento. Este campo tiene sentido cuando el campo Field vale VCRT_Field_Ex.

VCRT_Limit. Contiene información para un nivel de posiciones concreto.

Propiedad	Descripción
SymbolCode As String	Valor (código del valor) asociado.
LimitIndex As Integer	Nivel de la posición (1ª posición, 2ª…).
AskPrice As Double	Precio de venta de la posición del nivel LimitIndex.
AskVolume As Doubl	Volumen de venta de la posición del nivel LimitIndex.
AskOrders As Double	Número de órdenes de venta en la posición del nivel LimitIndex.
BidPrice As Double	Precio de compra de la posición del nivel LimitIndex
BidVolume As Double	Volumen de compra de la posición del nivel LimitIndex
BidOrders As Double	Número de órdenes de compra en la posición del nivel LimitIndex

Para recibir esta información es preciso invocar previamente el método **RequestSymbolFeed** con el parámetro Limits a TRUE. Una vez recibidas las posiciones, se pueden consultar estos valores en cualquier momento mediante los métodos **GetLimit/GetLimits**.

VCRT_RealTime. Es la interfaz principal de la librería. Desde aquí se configura ésta y se realizan las peticiones para obtener información de tiempo real.

Generalmente, la aplicación cliente crea un objeto de este tipo y define los eventos que quiere utilizar.

Propiedad	Descripción
TimerFrecuency As LongAs Long	Tiempo (milisegundos) máximo que se espera para enviar los ticks recibidos 1.
TicksBufferSize As Long	Tamaño máximo del buffer. Indica el número de ticks máximo que se desea acumular. Si se excede, se envían los ticks recibidos 1.

Cuando **VCRT_RealTime** recibe ticks, en lugar de enviarlos directamente al cliente, los acumula y posterga su envío. Esto supone una mejora en el rendimiento de la aplicación ya que disminuye considerablemente la mensajería entre el servidor COM y la aplicación cliente.

VCRT_RealTime tiene dos mecanismos que fuerzan el envío de la información acumulada:

TimerFrecuency: Cuando se excede el tiempo especificado en esta propiedad, se envían todos los ticks acumulados incluso si el buffer (cuyo tamaño se especifica en TicksBufferSize) no está lleno.

TicksBufferSize: Define el número de ticks que se acumulan como máximo. Si se excede dicho número de ticks, aunque no haya transcurrido el tiempo especificado en TimerFrecuency se envían los ticks al cliente.

Generalmente estos parámetros no se modifican o, en caso de hacerlo, se hace incrementando el valor de los mismos. Aplicaciones que no necesiten un refresco inmediato de los datos pueden incrementar estos valores para mejorar el rendimiento de sus aplicaciones.

***Importante**. En caso de disminuirse estos valores para recibir la información con menor retardo, es necesario tener presente que el envío de los ticks entre el servidor COM y la aplicación cliente consume tiempo, de modo que hay un punto en el cual, la disminución de estas propiedades se traduce en un mayor retardo, justo lo contrario de lo buscado al decrementar las citadas propiedades.*

EVENTOS

A continuación se detallan los distintos eventos que se pueden producir.

OnNewTicks (ArrayTicks() As VCRT_Tick). Proporciona acceso a los cambios que se han producido para un símbolo dado. Cada valor de ArrayTicks es un elemento VCRT_Tick con la información de un cambio en un campo indicado en el propio VCRT_Tick. Para recibir información debe haberse solicitado previamente mediante alguno de los métodos siguientes: RequestSymbolFeed, RequestSymbolsFeed, RequestFieldEx o RequestFieldsEx.
OnNewLimits (LimitsInfo() As VCRT_Limit). Proporciona acceso a los cambios que se han producido en las posiciones de un símbolo dado. Cada valor de LimitsInfo es un elemento VCRT_Limit con la información de la posición que ha cambiado. Para recibir información sobre las posiciones debe haberse solicitado previamente usando el método RequestSymbolFeed o RequestSymbolsFeed con el parámetro Limits a True. Este evento se produce después del evento OnLimitsChanged.
OnLimitsChanged (SymbolsChanged() As String). Notifica que se han producido cambios en las posiciones de los símbolos contenidos en SymbolsChanged. Los valores de las posiciones pueden consultarse usando los métodos GetLimit y GetLimits. Para recibir información sobre las posiciones debe haberse solicitado previamente usando el método RequestSymbolFeed o RequestSymbolsFeed con el parámetro Limits a True .Este evento se produce antes del evento OnNewLimits.
OnServerShutDown(). Este evento se ejecuta justo antes de que el servidor deje de estar disponible. Es útil para notificar a otras aplicaciones que ya no pueden usar el servidor o realizar tareas propias del cierre de la aplicación.

MÉTODOS

A continuación se detallan los métodos que se puede utilizar con los distintos objetos la librería VCRealTime.

RequestSymbolFeed (SymbolCode As String , Limits As Bool). Solicita la recepción en tiempo real de los cambios que se produzcan en el símbolo cuyo código se especifica en SymbolCode. Si Limits es True, se recibirá información de los cambios en las posiciones (eventos OnLimitsChanged y OnNewLimits) La información se recibe en el evento OnNewTicks y puede consultarse en cualquier momento mediante los métodos GetFieldText y GetFieldValue. Por cada solicitud que se haga y una vez deje de ser necesario recibir información de un símbolo dado, debe llamarse a CancelSymbolFeed.
RequestSymbolsFeed (Symbols() As String, Limits As Bool). Solicita la recepción en tiempo real de los cambios que se produzcan en los símbolos cuyos códigos se especifican en Symbols. La información se recibe del mismo modo que en RequestSymbolFeed.
CancelSymbolFeed (SymbolCode As String, Limits As Bool). Decrementa el número de solicitudes de tiempo real para el símbolo indicado. Cuando se llega a cero, se deja de recibir información en tiempo real de dicho símbolo. Es importante llamar a este método tan pronto como deje de ser necesaria la información del símbolo dado que se reduce el número de mensajes que debe procesar el servidor COM.
RequestSymbolFeed (SymbolCode As String , Limits As Bool) Solicita la recepción en tiempo real de los cambios que se produzcan en los símbolos cuyos códigos se especifican en Symbols. La información se recibe del mismo modo que en RequestSymbolFeed.
GetFieldText (SymbolCode As String, FieldType As enumVCRTField) As String. Devuelve el valor que tiene el símbolo SymbolCode para el campo especificado en enumVCRTField. Es necesario que se haya solicitado información para ese símbolo (ver RequestSymbolFeed o RequestSymbolsFeed) y haga llegado dicha información en el evento OnNewTicks.
GetFieldValue (SymbolCode As String, FieldType As enumVCRTField) As Double. Devuelve el valor numérico que tiene el símbolo SymbolCode para el campo indicado en enumVCRTField. *Nota: Para los campos extendidos, utilizar el método GetFieldExValue .*
GetLimit (SymbolCode As String, IdxLimit As Long) As VCRT_Limit. Devuelve un VCRT_Limit con la información sobre la IdxLimit -ésima posición del símbolo SymbolCode . Es necesario que se haya solicitado información para ese símbolo con el parámetro Limits a True (ver RequestSymbolFeed o RequestSymbolsFeed) y haga llegado dicha información en el evento OnNewTicks.
GetLimits (SymbolCode As String, LimitsInfo() As VCRT_Limit) As Long. Devuelve en LimitsInfo la misma información que GetLimit pero para todas las posiciones en lugar de solo para una dada. Devuelve el número de posiciones del símbolo.
FindSymbols (TextQuery As String, ArraySymbols() As String). Ejecuta una consulta de símbolos y devuelve en ArraySymbols los códigos de todos los que coincidan con el patrón especificado en TextQuery.
RequestFieldEx (SymbolCode As String, FieldEx As Long). Solicita la recepción en tiempo real de los cambios que se produzcan en el campo extendido FieldEx (ver enumVCRTExtField) del símbolo cuyo código se especifica en SymbolCode . La información se recibe en el evento OnNewTicks y puede consultarse en cualquier momento mediante los métodos GetFieldExText y GetFieldExValue.
RequestFieldsEx (SymbolCode As String, ArrayFields() As Long). Solicita la recepción en tiempo real de los cambios que se produzcan en los campos extendidos especificados en ArrayFields (ver enumVCRTExtField) del símbolo cuyo código se especifica en SymbolCode . La información se recibe del mismo modo que en RequestFieldEx.
GetFieldExText (SymbolCode As String, FieldEx As Long) As String. Devuelve el valor que tiene el símbolo SymbolCode para el campo extendido especificado en FieldEx . Es necesario que se haya solicitado información para ese símbolo (ver RequestFieldEx o RequestFieldsEx) y haga llegado dicha información en el eventoOnNewTicks.
GetFieldExValue (SymbolCode As String, FieldEx As Long) As Double. Devuelve el valor numérico que tiene el símbolo SymbolCode para el campo extendido especificado en FieldEx . Es necesario que se haya solicitado información para ese símbolo (ver RequestFieldEx o RequestFieldsEx) y haga llegado dicha información en el evento OnNewTicks.

EJEMPLO PRÁCTICO DEL USO DEL SERVIDOR VCREALTIME

En el ejemplo siguiente, vamos a ver cómo hacer uso del objeto **VCRT_Tick.** En este caso particular, se utilizará el evento **OnNewTicks**, el cual devuelve un array con los últimos elementos VCRT_Tick recibidos.

Cuando se recibe el evento, cada objeto VCRT_Tick que nos llega hace referencia a los datos propios de un tick (hora, ultimo, compra1, venta1, etc.

El ejemplo va a consistir en una hoja excel que reciba los datos Hora, Ultimo, Compra1, Venta1 y Volumen de un título en concreto (a elegir), y además, y que vaya actualizando dichos datos con cada nuevo tick que recibamos.

Los pasos se pueden resumir en los siguientes:

1º Preparar el escenario. Para ello, en una hoja de excel vacía, se define la interfaz a través de la cual se introducirá el código del símbolo y se visualizarán la información solicitada:

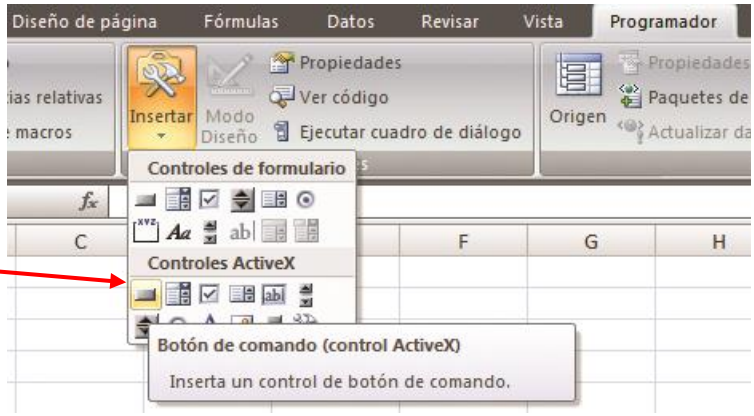
En la celda de **Simbolo del Titulo**, especificaremos en cada momento sobre qué activo queremos trabajar. Será una celda de entrada. El resto de celdas descritas serán celdas de salida y mostrarán los datos que recibamos desde Visual Chart V.

2º Generar evento para la descarga de información. Debemos de generar algún evento que ponga en marcha el proceso de descarga de datos. A continuación se detallan los pasos a seguir para crear un **botón de comandos** que nos sirva para dicho propósito. Los pasos a seguir son los siguientes:

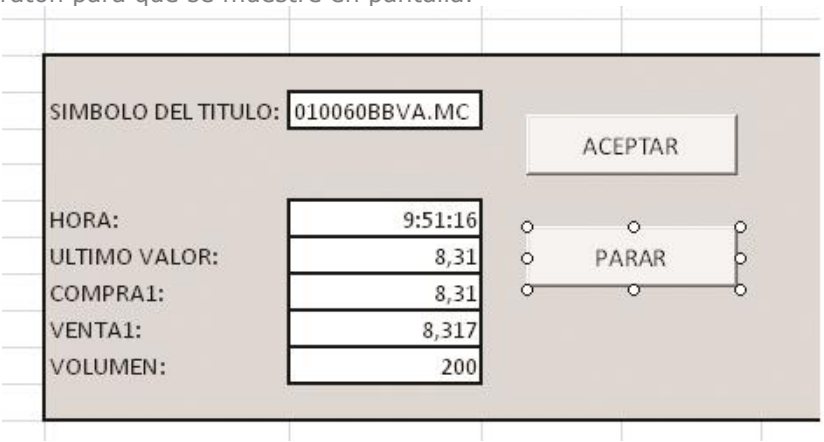
1. Activamos el Modo diseño, dentro del menú **Programador**.

2. Pulsamos sobre el menú **Insertar**

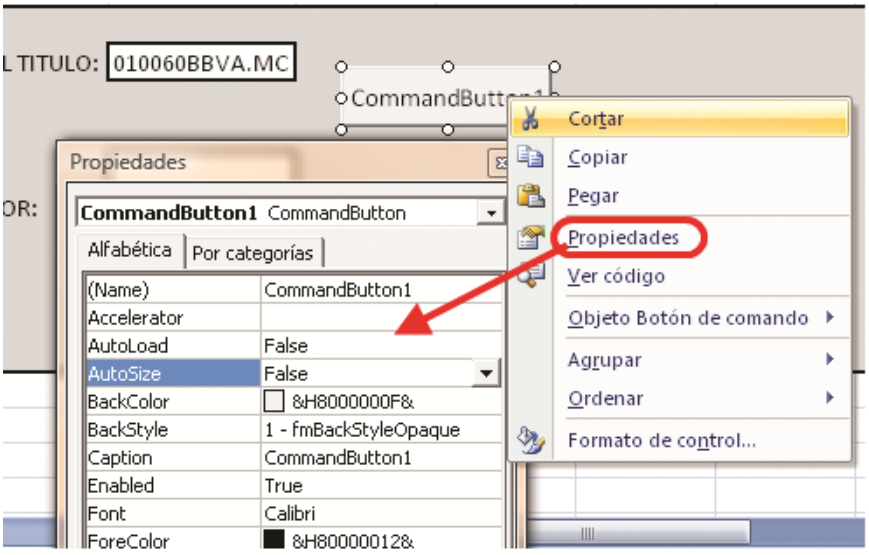
3. Hacemos clic sobre botón de comando (**Control Active X**).



4. Nos situamos en el lugar de la ventana en el que deseamos que aparezca este objeto de control, y haremos clic con el ratón para que se muestre en pantalla.



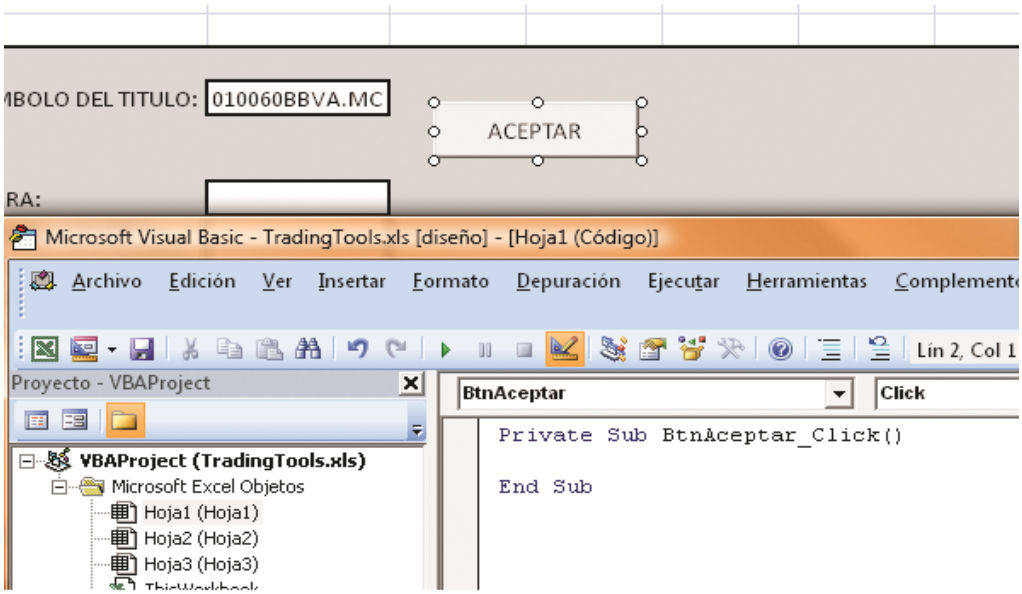
Para mejorar el aspecto, podemos pulsar sobre él con el botón derecho del ratón y acceder al editor de propiedades.



Desde este panel de propiedades modificaremos los parámetros correspondientes al nombre:

Name	**BtnAceptar**
Caption	ACEPTAR

5. Por último, hacemos doble clic sobre el botón de comando que acabamos de crear.



Como se puede ver en la imagen anterior, automáticamente se generará en el editor de Visual Basic el evento **Click** de nuestro botón.

De esta forma, cada vez que pulsemos sobre el botón **ACEPTAR**, se va a realizar lo que indiquemos dentro del evento, por lo tanto, aquí será donde iniciamos el código del programa.

***NOTA**: Antes de iniciar la programación, debemos confirmar que hemos añadido la referencia de la clase VCRealTimeLib. En caso de no hacerlo no podríamos usar los objetos y propiedades de ésta.*

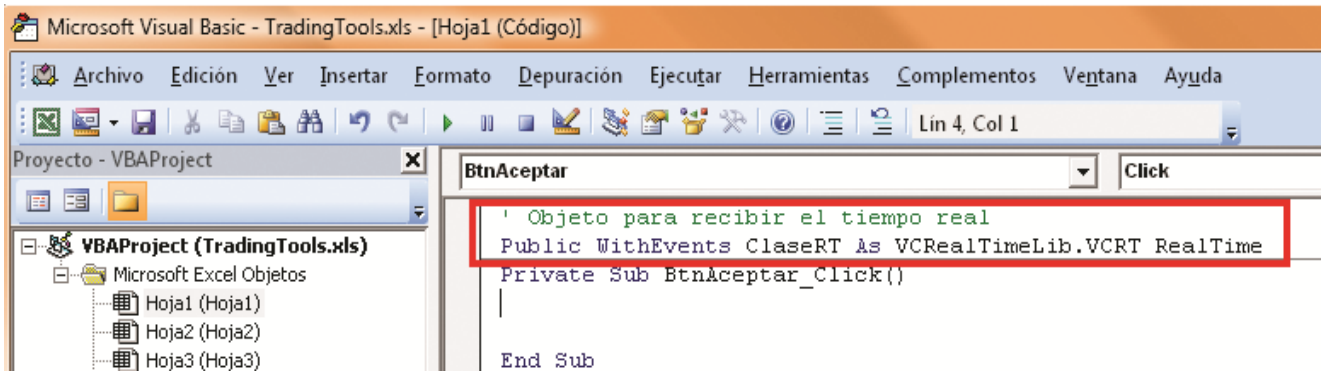
3º Programación de procedimientos

Como vamos a usar la clase VCRealTimeLib, definimos una variable que sea de esta clase y que llamaremos **ClaseRT**.

```
Public WithEvents ClaseRT As VCRealTimeLib.VCRT_RealTime
```

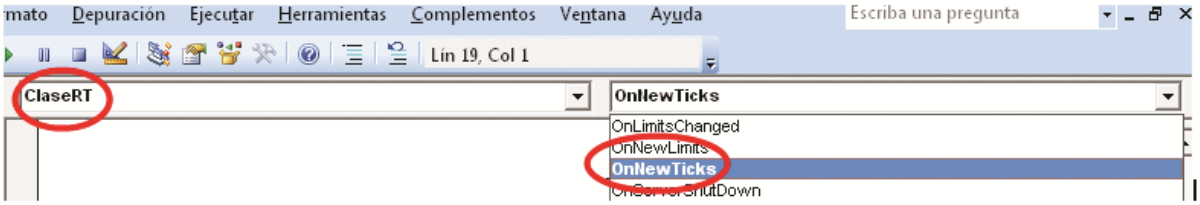
Un objeto que desencadena eventos se conoce como **origen del evento**.

Para controlar los eventos desencadenados por un origen de eventos, podemos declarar una variable de la clase del objeto con la palabra clave **WithEvents**. Esta palabra clave especifica que la variable **ClaseRT** se utilizará para controlar los eventos de un objeto de tipo **VCRT_RealTime**.



Declarada dicha variable, le indicamos al editor de VisualBasic que queremos generar el evento **OnNewTicks** haciendo lo siguiente:

En la pestaña de selección de objetos, escogemos el objeto **ClaseRT**. Automáticamente, en la pestaña de procedimientos, aparecerán los propios de este objeto. Del listado de procedimientos, seleccionamos el evento **OnNewTicks**.



En cuanto hagamos esto, se generará en nuestro código el evento **OnewTIcks** del objeto **ClaseRT**.

```
Private Sub ClaseRT_OnNewTicks(ArrayTicks() As VCRealTimeLib.VCRT_Tick)

End Sub
```

Este evento significa que cada vez que se genere un nuevo tick de los solicitados por el usuario, se va a realizar lo que indiquemos dentro de dicho procedimiento. En esta parte será donde iremos actualizando el contenido de las celdas declaradas en la preparación del escenario.

De este modo, ya tenemos dos procedimientos:

- Iniciar el programa
- Definir lo que iremos haciendo en cada ciclo, el cual estará determinado por la llegada de nuevos ticks.

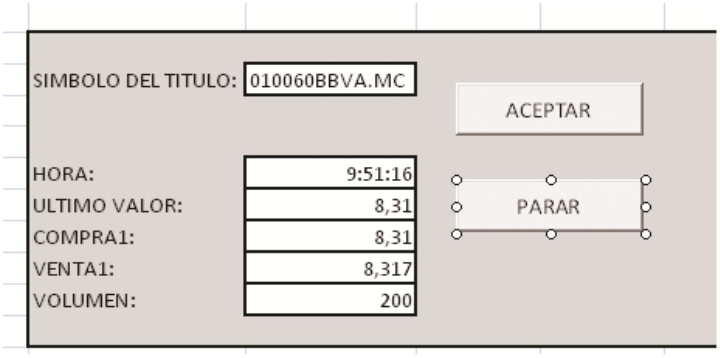
4º Generar evento para la detener la descarga de información

Solo queda determinar un modo de hacer que el programa finalice. Lo más sencillo es crear un nuevo botón de comando similar al anterior (BtnAceptar), pero que en este caso detendrá la descarga de datos.

Los pasos a seguir son los mismos que se han indicado en el 2º paso, aunque en esta ocasión, en las propiedades del objeto indicaremos lo siguiente:

Name	**BtnParar**
Caption	PARAR

Por tanto nuestra hoja excel va a quedar de la manera siguiente:



Tal y como hicimos con el botón **ACEPTAR**, pulsamos dos veces sobre el nuevo botón, y automáticamente se generará en el código el evento **Click**:

```
Private Sub BtnParar_Click()

End Sub
```

Para detener la descarga de datos, sólo tenemos que liberar la variable **ClaseRT**. Aprovecharemos también para inicializar el escenario.

Todos estos pasos los vamos a realizar desde un procedimiento nuevo, al cual llamaremos desde el evento **BtnParar_Click**.

A este nuevo procedimiento lo vamos a llamar **DetenerSistema** e incluye lo siguiente:

```
Public Sub DetenerSistema()
   Set ClaseRT = Nothing

   Range("D7:D11").ClearContents

   BtnAceptar.Enabled = True
   BtnParar.Enabled = False
End Sub
```

Para diferenciar visualmente cuándo está activo y desactivo el sistema, haremos lo siguiente:

- **Botón PARAR**. Vamos a desactivar este botón cuando no estemos descargando datos, y lo vamos a activar cuando sí lo estemos haciendo.

- **Botón ACEPTAR**. Vamos a desactivar este botón cuando estemos descargando datos, y lo vamos a activar cuando no lo estemos haciendo.

Lo siguiente será definir los pasos a seguir para poner en marcha el proceso de descarga. Esto lo haremos desde el procedimiento **BntAceptar_Click**. El código quedará de la siguiente manera:

```
Private Sub BtnAceptar_Click()
Dim Simbolo As String
   DetenerSistema

   Set ClaseRT = New VCRealTimeLib.VCRT_RealTime

   Simbolo = Range("D4")

   BtnAceptar.Enabled = False
   BtnParar.Enabled = True

   ClaseRT.TicksBufferSize = 20
   ClaseRT.RequestSymbolFeed Simbolo, False

End Sub
```

Lo que hemos hecho es lo siguiente:

1. Antes de nada, inicializar el escenario llamando al procedimiento **DetenerSistema**.

2. Crear un objeto **VCRT_RealTime** y conectar sus eventos con los procedimientos de evento asociados con **ClaseRT** mediante la declaración **"Set-New".**

3. Asignar a la variable **Simbolo** el código de símbolo definido en la celda D4.

4. Activar el botón **PARAR** y desactivar el botón **ACEPTAR**.

5. Definir el número de ticks que se van a acumular como máximo para forzar la llamada al **evento OnNewTicks**. Si se excede dicho número de ticks, aunque no haya transcurrido el tiempo especificado en **TimerFrecuency** se envían los ticks al cliente.

6. Solicitar la recepción en tiempo real de los cambios que se produzcan en el símbolo cuyo código se especifica en la variable **Simbolo**. Como el parámetro Limits es falso, no recibimos información de los cambios en las posiciones puesto que para nuestro ejemplo no los vamos a necesitar.

A partir de este momento, cada vez que pulsemos sobre el botón **ACEPTAR**, se creará un objeto **VCRT_RealTime** al cual le solicitamos información acerca del símbolo definido en la celda D4.

Los datos descargados podremos manipularlos en el evento **OnNewTicks** del modo siguiente:

```
Private Sub ClaseRT_OnNewTicks(ArrayTicks() As VCRealTimeLib.VCRT_Tick)
Dim i As Integer
   For i = 0 To UBound(ArrayTicks)
      Range("D7") = ArrayTicks(i).Date
      If ArrayTicks(i).Field = VCRT_Field_Last Then
         Range("D8") = ArrayTicks(i).Value
      Else If ArrayTicks(i).Field = VCRT_Field_Buy1 Then
         Range("D9") = ArrayTicks(i).Value
      Else If ArrayTicks(i).Field = VCRT_Field_Sale1 Then
         Range("D10") = ArrayTicks(i).Value
      Else If ArrayTicks(i).Field = VCRT_Field_Last_Vol Then
         Range("D11") = ArrayTicks(i).Value
      End If
   Next i
End Sub
```

Lo que hemos hecho es lo siguiente:

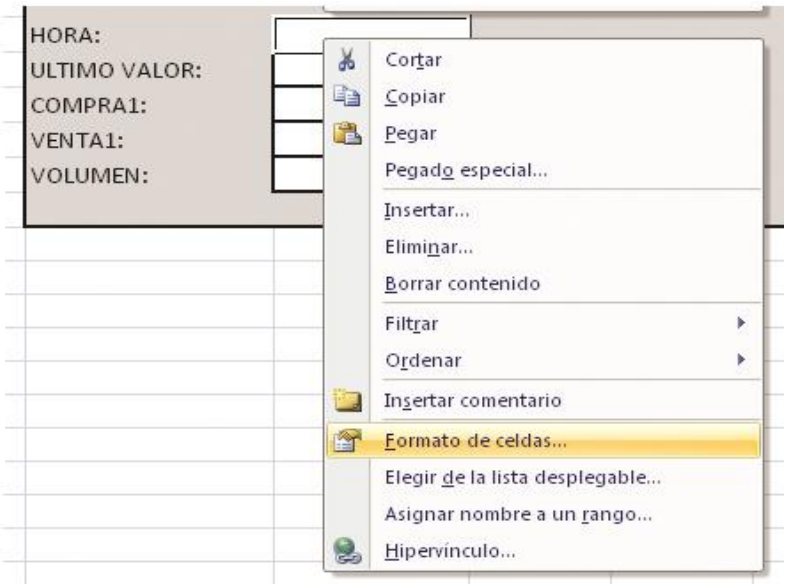
1. Como lo que recibimos es un array de ticks, lo recorremos desde la posición 0 hasta el final del array (UBOUND(ArrayTicks)

2. Al tener en cada posición del array un objeto **VCRT_Tick,** preguntarnos al objeto qué tipo de campo es (último, volumen, compra1, etc…), y según sea éste, guardaremos su valor (arrayticks(i).value) en la celda correspondiente.

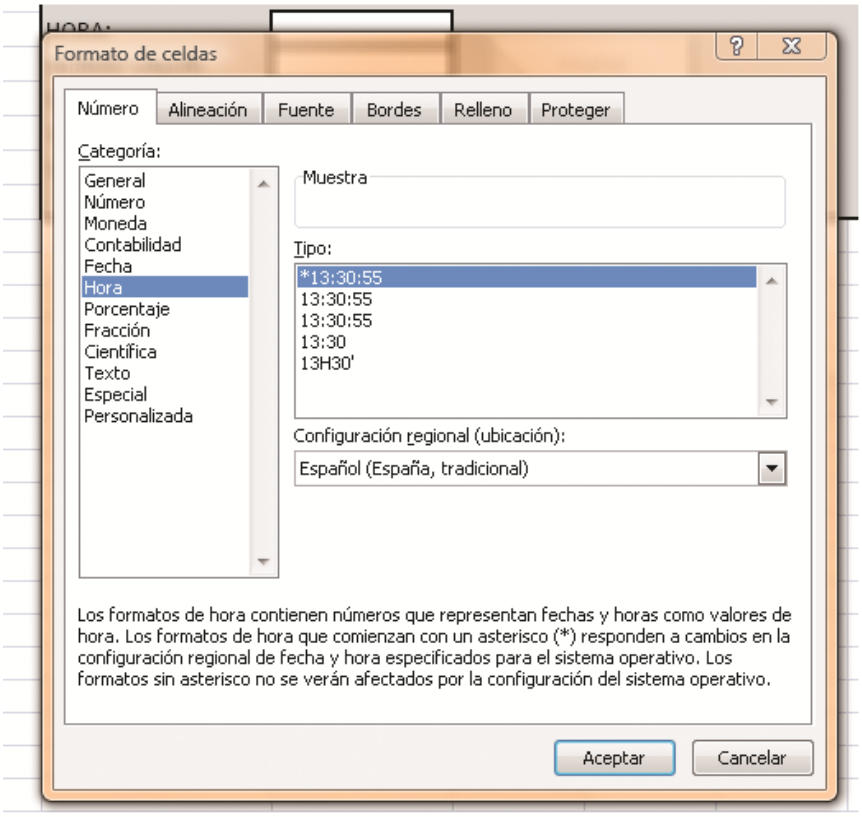
Para el caso del campo hora, no necesitamos preguntar qué tipo de campo es, ya que la propiedad fecha es común a todos los tipos de campo. Es por esto que, en este caso, escribiremos directamente en la celda de hora el valor de la propiedad fecha del último tick recibido (arrayticks(i).Date).

Puesto que sólo queremos saber la hora del último tick, y el formato de la propiedad **Date** es de tipo "dd/mm/yyyy hh:mm:ss", vamos a especificar para la celda de hora que el formato sea de tipo horario.

Para ello, seleccionamos con el botón derecho la celda D7 y elegimos la opción **Formato de celdas…**



En el cuadro de diálogo que se muestra a continuación, elegiremos la categoría **Hora** (dejando el tipo que aparece por defecto) y pulsaremos el botón **Aceptar.**



Por último, es necesario indicar que cuando se pulse el botón **PARAR** se llamará al procedimiento **DetenerSistema:**

```
Private Sub BtnParar_Click()
    DetenerSistema
End Sub
```

Hecho esto, tenemos terminado el proyecto y por tanto podemos cerrar el editor de VisualBasic.

Para finalizar desactivaremos el **Modo diseño** desde el menú **Programador** de Excel.

5º Visualizar la información

Antes de nada es necesario confirmar que en Visual Chart V estamos conectados al servidor de datos, y por otra parte, y que en nuestra licencia tenemos permisos para recibir información en tiempo real del activo que queramos probar.

A modo de ejemplo pediremos información en tiempo real del activo de BBVA. El código completo en Visual Chart para este valor es 010060BBVA.MC. Es importante tener en cuenta que **hay que indicar el código completo** de cualquier producto que deseemos usar.

Por norma general, el código de cada producto seguirá la siguiente nomenclatura:

010 + código del mercado + símbolo

Si no indicamos el código completo, Visual Chart no puede encontrar el valor, y por tanto, no enviará ningún dato a Excel.

Una vez hemos indicado el código en la celda D4, pulsaremos **ACEPTAR**. En ese momento empezaremos a recibir datos en tiempo real del activo.

Para dejar de recibir la información pulsaremos **PARAR.** De esta forma el vínculo entre Visual Chart y Excel se habrá roto y podremos tranquilamente cerrar nuestra hoja Excel.

VCDataSource

INTRODUCCIÓN

La librería **VCDataSource** proporciona acceso a la fuente de datos de:

- históricos de cotizaciones
- Indicadores
- Sistemas
- Valores estadísticos

De este modo, permite la consulta de cualquier clase de dato asociado a estos elementos, ya sea información sobre un símbolo determinado, por ejemplo el mínimo movimiento, diferencia horaria, etc., o el valor de un indicador en un momento concreto del histórico, las posiciones de compra y venta de o fiabilidad de un sistema, etc.

Como se ve en la imagen, el servidor actúa como intermediario entre Visual Chart y la aplicación cliente (por ejemplo Microsoft Excel).

El objeto principal es **VCDS_DataSourceManager**. Disponiendo de éste, es posible crear series de datos, indicadores, sistemas y recibir eventos cuando estos cambian.

Para poder hacer uso de éste servidor, la aplicación cliente debe de tener incorporada a su lista de referencias la clase **VisualChart Source Library.dll**

Al igual que ocurre con la librería de tiempo real, dispone de una colección de objetos, eventos y métodos que podemos usar para la manipulación de datos.

OBJETOS

Los objetos de la librería **VCDataSource** son los siguientes:

- VCDS_BarValue
- VCDS_TradeSession
- VCDS_SymbolInfo
- VCDS_Parameter
- VCDS_CatalogItem
- VCDS_SystemOrder
- VCDS_Trade
- VCDS_DataSource
- VCDS_DataSerie
- VCDS_System
- VCDS_Indicator
- VCDS_Context
- VCDS_DataSourceManager

VCDS_BarValue**.** Contiene los valores de una barra de un objeto **VCDS_DataSerie**.

Propiedad	Descripción
High As Double	Máximo de la barra.
Low As Double	Mínimo de la barra.
Close As Double	Cierre.
Open As Double	Apertura.
Date As Date	Fecha.
Volume As Double	Volumen de la barra.
OpenInterest As Double	Interés abierto.

VCDS_TradeSession. Define un periodo de una sesión (ver VCDS_SymbolInfo).

Propiedad	Descripción
OpenTime As Date	Fecha inicial del periodo.
CloseTime As Date	Fecha final del periodo.
DayOfWeek As Long	Día de la semana (Domingo = 1… Sábado = 7).

VCDS_SymbolInfo. Contiene información sobre un símbolo (ver GetSymbolInfo de VCDS_DataSource).

Propiedad	Descripción
Type As enumVCDSInstrumentType	Tipo de instrumento.
TradingSessions As VCDS_TradeSessions	Colección con las sesiones en que cotiza el valor.
Description As String	Descripción del valor.
Decimals As Long	Número de decimales.
TimeOffset As Long	Diferencia horaria.
PointValue As Double	Valor del punto.
MinMovement As Double	Mínimo movimiento.

VCDS_Parameter. Contiene información sobre un parámetro de un indicador o sistema.

Propiedad	Descripción
Name As String	Nombre del parámetro.
Description As String	Descripción.
CurrentValue As Object	Valor actual del parámetro.
DefaultValue As Object	Valor por defecto para el parámetro.
HighestLimit As Double	Valor máximo que puede tomar el parámetro.
LowestLimit As Double	Valor mínimo que puede tomar el parámetro.
ReadOnly As Bool	Indica si el parámetro es de solo lectura.

VCDS_CatalogItem. Contiene información sobre un indicador o sistema dado.

Propiedad	Descripción
Code As String	Código del indicador/sistema.
Name As String	Nombre del indicador/sistema.
Description As String	Descripción.
ParamsCount As Long	Número de parámetros del indicador/sistema.

VCDS_SystemOrder. Contiene información sobre una orden de un sistema (ver evento OnFilledOrders).

Propiedad	Descripción
OrderType As enumVCDSOrderType	Tipo de orden (ver enumVCDSOrderType).
Bar As Long	Barra de la orden.
Volume As Long	Volumen de la orden.
CummVolume As Long	Volumen acumulado.
IsLiquidate As Bool	Indica si se trata de una liquidación.
Price As Double	Precio de la orden.
Label As String	Etiqueta.
Commission As Double	Comisión.
PointValue As Double	Valor del punto.
SymbolCode As String	Código del símbolo.
Date As Date	Fecha de la orden.
Guarantee As Double	Garantía.
Balance As Double	Saldo.

VCDS_Trade. Contiene información sobre un negocio de un sistema.

Propiedad	Descripción
EntryPrice As Double	Precio de entrada del negocio.
ExitPrice As Double	Precio de salida del negocio.
EntryDate As Date	Fecha de entrada del negocio.
ExitDate As Date	Fecha de salida del negocio.
Side As enumVCDSOrderSide	Indica si el negocio se debe a una compra o una venta.
OrderType As enumVCDSOrderType	Tipo de orden que generó el negocio.
PercentNetProfit As Double	Ganancia porcentual del negocio.
NetProfit As Double	Ganancia del negocio.
EntryCommission As Double	Comisión de entrada.
ExitCommission As Double	Comisión de salida.
InitContracts As Object	Número de contratos del negocio.
RemainContracts As Object	Número total de contratos comprados o vendidos en el sistema cuando se ha producido el negocio.
Label As String	Etiqueta.
IsCurrentOpenPosition As Bool	Indica si se trata de un negocio ejecutado en el sistema (False) o del negocio que se forma si se cerrase la

	posición abierta actual (True).

VCDS_DataSource. VCDS_DataSource es una interfaz que implementan todas las series de datos y provee de información básica sobre las mismas.

Propiedad	Descripción
Id As Long	Identificador de la serie. Todas las series tienen un valor numérico asociado que las identifica de forma unívoca.
Code As String	Código de la serie.
Name As String	Nombre de la serie.
Compression As Long	Compresión (unidades).
CompressionType As enumVCDSCompressionType	Tipo de compression.
ActiveEvents As Bool	Indica si se desea recibir eventos de la serie 1.
Size As Long	Número de elementos que tiene la serie.
Type As enumVCDSDataSourceType	Tipo de serie (indicador, sistema…).
InitDate As Date	Fecha inicial de la serie.
EndDate As Date	Fecha final de la serie.

Si no es necesario recibir los eventos de una serie, es conveniente desactivarlos ya que se mejora el rendimiento de la aplicación.

El valor inicial que toma esta propiedad cuando se crea la serie, es el mismo que tiene la propiedad **ActiveEvents** del objeto **VCDS_DataSourceManager**.

VCDS_DataSerie. Define una serie de datos y provee acceso a los distintos valores de la barra del símbolo (apertura, mínimo, volumen…). Implementa la interfaz IVCDS_DataSource.

Las series se crean con el método NewDataSerie de VCDS_DataSourceManager. Cada vez que se genera una nueva barra en la serie se dispara el evento OnNewDataSerieBar de VCDS_DataSourceManager.

VCDS_System. Representa un sistema y provee acceso a los negocios del mismo así como a valores estadísticos del sistema.

Durante el cálculo del sistema, los eventos OnSystemEvent y OnFilledOrders notifican los cambios de posición, las ejecuciones de órdenes y la generación de negocios.

Los sistemas se crean con el método NewSystem de VCDS_DataSourceManager.

Propiedad	Descripción
ParentSource As VCDS_DataSource	Indica la serie padre del sistema.
StartBar As Long	Indica en que barra empieza a calcularse el sistema.
MarketPosition([Index As Object]) As numVCDSMarketPosition	Indica la posición del sistema en el Index-ésimo negocio. Sin parámetros devuelve la posición actual del sistema.

VCDS_Indicator. Representa un indicador y provee acceso a los distintos valores de sus líneas. El evento OnNewIndicatorBar se dispara cada vez que se crea una nueva barra en el indicador.

Los indicadores se crean con el método NewIndicator de VCDS_DataSourceManager.

Propiedad	Descripción
ParentSource As VCDS_DataSource	Indica la serie padre del indicador.
StartBar As Long	Indica en que barra empieza a calcularse el indicador.
NumberOfLines As Long	Número de líneas del indicador. Este valor puede ser cero si el indicador no se ha calculado todavía.

VCDS_Context. Los contextos actúan como un contenedor de series (objetos que implementan la interfaz VCDS_DataSource) que comparten la escala temporal.

Las series tienen que estar contenidas forzosamente en un contexto. Si no se define un contexto, se les asigna uno de forma automática. Cuando un contexto se queda sin series, se elimina automáticamente.

Los contextos que contienen series de ticks, sólo pueden contener un único DataSerie. Las series de ticks no pueden compartir contexto.

No hay restricciones con el resto de tipos de series, indicadores y sistemas, que siempre comparten el contexto de su serie padre.

Cuando se crea una serie (ver NewDataSerie de VCDS_DataSourceManager) se le puede indicar el contexto que se desea para la nueva serie. Dicho contexto puede obtenerse de la propiedad Context de un VCDS_DataSource creado previamente.

La propiedad ContextWorkMode de VCDS_DataSourceManager especifica el modo de actuación en caso de no indicar un contexto en la creación de series:

VCDS_CWM_Multiple: La serie se creará en un nuevo contexto.
VCDS_CWM_Shared: La serie se creará dentro del contexto por defecto (ver enumVCDSContextWorkMode).

Propiedad	Descripción
Id	Identificador del contexto. Todos los contextos tienen un valor numérico asociado que lo identifica de forma unívoca.
Name	Nombre del contexto.

VCDS_DataSourceManager. Este objeto representa la interfaz principal del servidor COM y genera los eventos de las distintas series que pueden crearse.

Propiedad	Descripción
ContextWorkMode	Modo de trabajo del contexto. Define como se actúa cuando se crean series sin indicar el contexto en que deben residir (ver enumVCDSContextWorkMode).
StoreHistoricSystemOrder	Permitir que el usuario elija si desea guardar o no el histórico de órdenes lanzadas por el sistema.

EVENTOS

El servidor **VCDataSource** proporciona amplia información de los datos resultantes en el histórico de la fuente de datos. Además, puesto que dichas fuentes de datos son tablas dinámicas, también permite obtener los datos que se van generando durante el tiempo real.
Existe por tanto una opción de notificación de nuevos datos mediante la recepción de eventos. Cada vez que se genera un nuevo valor, el cliente recibe un evento con la información resultante. Existen eventos para todas las clases de objetos disponibles.

A continuación se detallan los distintos eventos que se pueden producir.

OnNewDataSerieBar (DataSerie As VCDS_DataSerie). Ocurre cuando se crea una nueva barra en la serie.
OnNewIndicatorBar (Indicator As VCDS_Indicator). Ocurre cuando se crea una nueva barra en el indicador.
OnSystemEvent (System As VCDS_System, EventType As enumVCDSSystemEvent). Ocurre cuando hay una nueva orden o negocio y cuando se cambia de posición. EventType indica que evento es el que se ha producido.
OnFilledOrders (System As VCDS_System, Orders As VCDS_SystemOrders). Ocurre cuando se ejecuta una orden en el sistema System. Orders contiene una lista con las órdenes que se han ejecutado. Se dispara inmediatamente después de la ejecución de una orden.
OnServerShutDown (). Este evento se ejecuta justo antes de que el servidor deje de estar disponible. Es útil para notificar a otras aplicaciones que ya no pueden usar el servidor o realizar tareas propias del cierre de la aplicación.

MÉTODOS

Si queremos extraer datos de cada uno de los distintos tipos de objetos propios del servidor **VCDataSource**, debemos hacer uso de los métodos asociados a cada clase de objeto.

A continuación se detallan los métodos que se puede utilizar con los distintos objetos de la librería **VCDataSource.**

VCDS_CatalogItem

GetParam(Index As Object) As VCDS_Parameter. Devuelve el parámetro indicado por Index. Index puede hacer referencia a la posición del parámetro (si se pasa un valor numérico) o bien contener el nombre del parámetro (si se pasa un String)
GetParameters() As VCDS_Parameters. Devuelve una colección con todos los parámetros del indicador/sistema

VCDS_DataSource

Context() As VCDS_Context. Devuelve el contexto de la serie (ver VCDS_Context).
GetSymbolInfo () As VCDS_SymbolInfo. Devuelve información sobre el símbolo de la serie.
Indicators() As VCDS_Indicators. Devuelve una colección con los indicadores de la serie.

VCDS_DataSerie

GetBarValues(Index As Object) As VCDS_BarValue. Devuelve la Index-ésima barra de la serie.
GetBarsValues(InitIndex As Object, EndIndex As Object) As VCDS_BarValue(). Devuelve un array con las barras comprendidas entre InitIndex y EndIndex . Ambos parámetros puede ser una fecha de la serie o una posición entre 1 y Size.
Systems() As VCDS_Systems. Devuelve una colección con los sistemas de la serie.

VCDS_System

GetCalculateOptions(Unit As enumVCDSStatisticMeasurementUnit, CompType As enumVCDSStatisticCompression, Compression As Long, BeginDate As Date, EndDate As Date). Devuelve la configuración actual de la estadística del sistema 1.

SetCalculateOptions(Unit As enumVCDSStatisticMeasurementUnit, CompType As enumVCDSStatisticCompression, Compression As Long, BeginDate As Date, EndDate As Date). Asigna los valores con los que se harán los cálculos de la estadística 1.

GetStatisticVariableSize(Variable As enumVCDSStatisticVariable) As Long. Indica el número de elementos que tiene la variable estadística indicada.

GetStatisticVariableDate(Variable As enumVCDSStatisticVariable, pVal As Long) As Date. Indica la fecha asociada al valor pVal-ésimo de la variable estadística indicada.

GetStatisticVariableValue(Variable As enumVCDSStatisticVariable, Index As Long) As Object. Devuelve el Index-ésimo valor de la variable estadística indicada.

GetStatisticVariableValues(Variable As enumVCDSStatisticVariable) As Object(). Devuelve un array con todos los valores de la variable estadística indicada.

GetTrade([Index As Object]) As VCDS_Trade Devuelve el Index-ésimo negocio del sistema.

GetTradeRange([InitIndex As Objetc], [EndIndex As Objetc]) As VCDS_Trades. Devuelve una colección con todos los negocios del sistema comprendidos entre InitIndex y EndIndex . Ambos parámetros pueden referirse a una posición o una fecha del negocio. Si se omite InitIndex , se comienza desde el primer negocio. Si se omite EndIndex , se termina en el último negocio.

GetParameters() As VCDS_Parameters. Devuelve una colección con los parámetros del sistema.

Setting. Es el equivalente a modificar los campos de **Ajustes** dentro de las propiedades de un sistema.

De este modo, si deseamos cambiar alguno de los ajustes del objeto VCDS_System se procede del siguiente modo:

1. Primero, se especifican qué campos del método Setiing se desean cambiar.

```
'-- Especificar que vamos a añadirle comisión y deslizamiento por puntos al sistema
Sistema.Settings.AgencyCommision = 2
Sistema.Settings.Slippages = 1
Sistema.Settings.PorcentualPenalty = False
```

2. Segundo, para que el cambio en los ajustes tenga efecto, llamamos a la sentencia Apply del método:

```
'-- Especificar que vamos a añadirle comisión y deslizamiento por puntos al sistema
Sistema.Settings.AgencyCommision = 2
Sistema.Settings.Slippages = 1
Sistema.Settings.PorcentualPenalty = False
```

La relación entre los atributos de la sección de **Ajustes** y los campos del método **Settings** es la siguiente:

Slippages (Sistema.Settings.Slippages)
Comisión Agencia (Sistema.Settings.AgencyCommision)
Modo de penalización (Sistema.Settings.PorcentualPenalty)
Método de entrada (Sistema.Settings.InputMode)
Modo de Liquidación (Sistema.Settings.CancelPosAfterLiquidate)
Núm. Max. Etiquetas (Sistema.Settings.LabelLimit)
Núm. Max. Contratos/Acciones (Sistema.Settings.StockLimit)
Ejecutar órdenes limitadas (Sistema.Settings.ExecModeLimited)
Presupuesto (Sistema.Settings.Budget)
Cantidad Presupuesto (Se activa si el Presupuesto no vale 0)
Valor por Punto (Sistema.Settings.PointValue)

La estadística se configura con unos parámetros dados y todos los resultados que se devuelven en el resto de métodos, lo hacen con esas configuraciones.

VCDS_Indicator

Value(Index As Object, [Line As Object]) As Double. Devuelve el Index-ésimo valor del indicador para la línea Line . Si se omite el parámetro Line se usa la primera línea. Index puede ser una posición o una fecha.
Values(BeginIndex As Object, EndIndex As Object, [Line As Object]) As Oject(). Devuelve un array con los valores del indicador comprendidos entre BeginIndex y EndIndex para la línea indicada (la primera si se omite el parámetro Line). Los índices pueden ser una posición o una fecha.
GetPosition([Index As Object], [Line As Object]) As enumVCDSMarketPosition. Devuelve la posición alcista/bajista/neutra del indicador para la posición y línea indicadas. Se usa la primera línea en caso de omitir el parámetro Line . Si se omite el parámetro Index , se devuelve la posición actual del indicador.
GetLineName(Line As Long) As String. Devuelve el nombre de la línea Line del indicador.
GetParameters() As VCDS_Parameters. Devuelve una colección con los parámetros del indicador.

VCDS_Context

DataSeries() As VCDS_DataSeries. Devuelve una colección con todas las series que contiene el contexto.
Systems() As VCDS_Systems. Devuelve una colección con todos los sistemas que contiene el contexto.
Indicators() as VCDS_Indicators. Devuelve una colección con todos los indicadores que contiene el contexto.

VCDS_DataSourceManager

Contexts() As VCDS_Contexts. Devuelve una colección con todos los contextos existentes.
DataSeries() As VCDS_DataSeries. Devuelve una colección con todas las series existentes.
Systems() As VCDS_Systems. Devuelve una colección con todos los sistemas existentes.
Indicators() As VCDS_Indicators. Devuelve una colección con todos los indicadores existentes.
NewDataSerie (SymbolCode As String, CompType As enumVCDSCompressionType, CompressionUnits As Long, [InitDate As Object], [EndDate As Object], [Context As Object]) As VCDS_DataSerie. Crea una nueva serie de datos para el símbolo cuyo código está especificado en SymbolCode . CompType y CompressionUnits especifican el tipo y unidades de compresión con que se creará la serie. Las fechas permiten especificar el rango de serie que se desea cargar (si se omiten se cogen los valores por defecto que usa VisualChart). El contexto permite crear una serie para que utilice el mismo contexto que otra ya creada.
NewIndicator (Name As String, DataSource As VCDS_DataSource, Parameters As Object()) As VCDS_Indicator. Crea el indicador especificado en Name sobre la serie DataSource usando los parámetros especificados.
NewSystem (Name As String, DataSource As VCDS_DataSource, Parameters As Object()) As VCDS_System. Crea el sistema especificado en Name sobre la serie DataSource usando los parámetros especificados.
DeleteDataSource(DataSource As VCDS_DataSource). Elimina la serie indicada.
DeleteAll(). Elimina todas las series y contextos.
GetIndicatorInfo(Name As Object) As VCDS_CatalogItem. Devuelve la información asociada al indicador Name (ver VCDS_CatalogItem).
GetSystemInfo(Name As Object) As VCDS_CatalogItem. Devuelve la información asociada al sistema Name (ver VCDS_CatalogItem).
GetIndicatorCatalog() As VCDS_CatalogItems. Devuelve una colección con la información asociada a todos los indicadores (ver VCDS_CatalogItem).
GetSystemCatalog() As VCDS_CatalogItems. Devuelve una colección con la información asociada a todos los sistemas (ver VCDS_CatalogItem).

GetDataSource(Id As Long) As VCDS_DataSource. Devuelve un objeto serie (VCDS_DataSerie , VCDS_Indicator o VCDS_System) dado su Id.

COLECCIONES

Todas las colecciones tienen la misma interfaz y el mismo funcionamiento. Se utilizan para devolver un conjunto de datos del mismo tipo.

La siguiente tabla muestra las colecciones existentes y el tipo de objeto que contienen:

Colección	Tipo de objeto
VCDS_TradeSessions	VCDS_TradeSession
VCDS_Parameters	VCDS_Parameter
VCDS_CatalogItems	VCDS_CatalogItem
VCDS_SystemOrders	VCDS_SystemOrder
VCDS_Trades	VCDS_Trade
VCDS_DataSources	VCDS_DataSource
VCDS_DataSeries	VCDS_DataSerie
VCDS_Systems	VCDS_System
VCDS_Indicators	VCDS_Indicator
VCDS_Contexts	VCDS_Context

Propiedad	Descripción
Count As Long	Devuelve el número de elementos de la colección.
Item(index As Object) As XXX	Devuelve un ítem de la colección (XXX se refiere al tipo de la colección mostrado en la segunda columna de la tabla anterior)
_NewEnum	Todas las colecciones implementan este método oculto para permitir el recorrido por los distintos ítems de la colección usando foreach.

EJEMPLO PRÁCTICO DEL USO DEL SERVIDOR VCDATASOURCE

En el ejemplo que veremos a continuación, vamos a buscar la diferencia porcentual entre la apertura y cierre de las barras, de un histórico determinado, para un valor concreto. Además de este dato, queremos saber la tendencia que marca el RSI en cada momento, según la siguiente regla:

- Cuando el RSI venga de sobreventa (cruza desde abajo el 30) marcará tendencia alcista.
- Cuando el RSI venga de sobrecompra (cruza desde arriba el 70) marcará tendencia bajista.

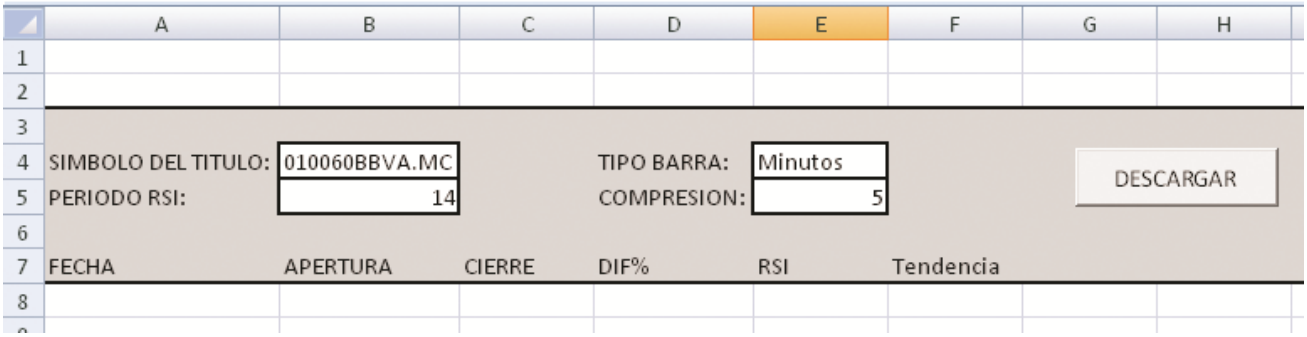
Una vez más, vamos a usar como aplicación cliente Microsoft Excel. Aprovecharemos el libro Trading Tools que ya hemos creado en el ejemplo anterior, y trabajaremos en la hoja 2 de éste, puesto que en la hoja 1 tenemos el ejemplo previo de tiempo real.

*Es necesario tener en cuenta que puesto que vamos a hacer uso del objeto **VCDS_DataSourceManager** es necesario haber añadido a la lista de referencias la librería VisualChart Source Library 1.0*

Los pasos a seguir se pueden resumir en los siguientes:

1º Preparación del escenario

Lo primero que haremos será diseñar la interfaz visual para introducir y mostrar la información. La hoja 2 de nuestro libro Excel debe quedar de la siguiente manera:



2º Generar evento para la descarga de información

El botón **DESCARGAR** es un botón de comando similar a los botones ACEPTAR y PARAR de la hoja ejemplo de tiempo real.

Recordar que para crear controles en una hoja Excel, es necesario seleccionar el menú de **Programador**, activar la opción **Modo Diseño**, y posteriormente seleccionar la opción **Insertar → Botón de comando (active X).**

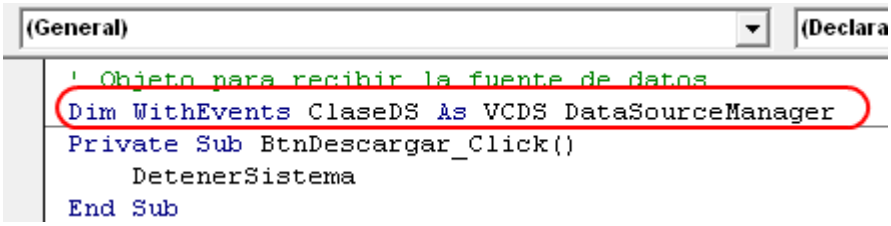
Como hicimos con los otros dos botones del ejemplo anterior, accedemos al panel de propiedades del objeto y desde éste modificamos los parámetros correspondientes al nombre:

Name	**BtnDescarga**
Caption	DESCARGAR

A continuación, pulsamos dos veces sobre el botón para que se genere el evento **Click** en el editor de Visual Basic. Recordemos que este evento nos indica que cada vez que el usuario pulse sobre el botón **DESCARGAR**, se llevará a cabo lo que indiquemos dentro del evento.

3º Programación de procedimientos. Tal y como se ha comentado con anterioridad, la clase principal del servidor **VCDataSource** es la clase **VCDS_DataSourceManager**, por lo que vamos a crear un objeto llamado **ClaseDS** que sea de este tipo:

```
Public WithEvents ClaseDS As VDS_DataSourceManager
```



Además de este objeto, crearemos dos objetos más, uno que será de tipo fuente de datos, y otro que será de tipo indicador, puesto que son los dos tipos de objetos que necesitamos.

```
Dim Fuente As VCDS_DataSource
Dim RSIData As VCDS_Indicator
```

El objeto **RSIData** lo hemos llamado así puesto que siempre vamos a hacer referencia al indicador RSI para esta clase de objeto en concreto.

Como vamos a ir guardando el valor de la tendencia, también crearemos una variable global llamada **TendenciaRSI**, de modo que podrá valer 1 si la tendencia es alcista y -1 si es bajista.

```
Dim TendenciaRSI As Integer
```

La estructura del programa va a constar de tres partes:

- Método de inicio y creación de objetos → Usaremos el evento BtnDescargar_Click.
- Método de descarga de datos → Usaremos un procedimiento nuevo que llamaremos IniciarDescarga.
- Método de finalización del programa → Usaremos un procedimiento nuevo que llamaremos DetenerSistema.

El procedimiento **DetenerSistema** servirá para reiniciar el escenario y para liberar a los distintos objetos que hemos creado durante el proceso. El código quedará de la siguiente manera:

```
Public Sub DetenerSistema()
    Set Fuente = Nothing
    Set RSIData = Nothing

    If Not ClaseDS Is Nothing Then
        ClaseDS.DeleteAll
        Set ClaseDS = Nothing
    End If

    BtnAceptar.Enabled = True
    BtnParar.Enabled = False
End Sub
```

Lo que hemos hecho es lo siguiente:

1. Liberar los objetos utilizados.
2. Con el método **DeleteAll** lo que hacemos es eliminar todos los vínculos de fuentes que pudiera mantener la aplicación cliente con Visual Chart.
3. Activar el botón **DESCARGAR**

En cuanto al código de inicio y creación de objetos, quedará de la siguiente manera:

```
Private Sub BtnDescargar_Click()
Dim Simbolo As String
Dim MiComp As enumVCDSCompressionType
Dim MiCompUni As Integer
Dim PeriodoRSI As Integer
On Error GoTo Fallo

    Range("A8:F8000").ClearContents
    Range("A8:F8000").ClearFormats
```

```
    DetenerSistema
    '..
    Set ClaseDS = New VCDS_DataSourceManager

    BtnDescargar.Enabled = False

    Simbolo = Range("B4")

    'crear fuente
    If Simbolo <> "" Then
       MiCompUni = CInt(Range("E5"))
       If Range("E4") = "Minutos" Then MiComp = VCDS_CT_Minutes
       If Range("E4") = "Dias" Then MiComp = VCDS_CT_Days
       Set Fuente = ClaseDS.NewDataSerie(Simbolo, MiComp, MiCompUni)
       'confirmar que se ha creado
       If Fuente.Size > 0 Then
          PeriodoRSI = Range("B5")
          TendenciaRSI = 0
          'crear indicador RSI
          Set RSIData = ClaseDS.NewIndicator("RSI", Fuente, PeriodoRSI, 70, 30)
          'confirmar que se ha creado el RSI
          If RSIData.Size > 0 Then
          IniciarDescarga
          Else
            DetenerSitema
      End If
Else
          DetenerSistema
       End If
    End If
  Fallo:
     If Err.Number <> 0 Then DetenerSistema
  End Sub
```

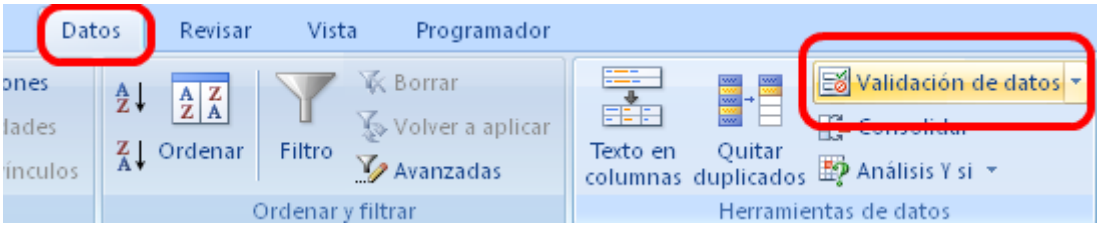
En este procedimiento hemos declarado varias variables locales:

Simbolo: Donde guardaremos el código de símbolo del valor a analizar.

MiComp: Esta variable es de tipo enumVCDSCompressionType. Este tipo de enumeración pertenece a la librería VCDataSource y sirve para definir los distintos tipos de compresión posibles (minutos, días, semanas, etc…). Para nuestro ejemplo, vamos a permitir que sea minutos o días según lo que ponga en la celda TIPO BARRA. Si en la celda indicamos "Dias", entonces MiComp será de tipo VCDS_CT_Days y si indicamos "Minutos", entonces MiComp será de tipo VCDS_CT_Minutes.

Si queremos darle mayor calidad a nuestra interfaz, podemos incluso obligar a que en dicha celda sólo se puedan especificar esos dos valores. Para esto:

1. Escribimos en las celdas F4 y F5 los textos "Minutos" y "Dias" respectivamente.
2. Cambiamos el color de fuente de estas dos celdas para que tengan el mismo color que el fondo (de este modo no se verán).
3. Seleccionamos la celda E4 y elegimos la opción **Validación de datos** del menú **Datos**:



4. En el cuadro de diálogoseleccionamos entre las opciones de Permitir, la opción **Lista**, y en **Origen** escribimos "=$F$4:$F$5".
5. Por último pulsamos **Aceptar**.

MiCompUni: Donde guardaremos el número de unidades de la compresión de barras. Es decir, el contenido de la celda E5.

PeriodoRSI: Donde guardaremos el periodo del indicador, es decir, el contenido de la celda B5.

En el procedimiento, hemos hecho lo siguiente:

1. Inicializar el escenario limpiando el contenido de las celdas. Para ello usamos las propiedades del rango de celdas ClearFormats y ClearContents.
2. Llamar al método **DetenerSistema** para liberar previamente a todos los objetos que usaremos.
3. Crear el objeto **ClaseDS**.
4. Inhabilitar el botón **DESCARGAR** para que tengamos constancia de que mientras descargamos datos no podemos volver a pulsarlo.
5. Una vez confirmamos que Simbolo contiene información, creamos primero el objeto Fuente usando el método de la clase ClaseDS **NewDataSerie**. Este método nos pide el código del simbolo y la compresión de barras.
6. Una vez hemos declarado **NewDataSerie** pueden pasar tres cosas:
    a. Que se produzca un error porque o bien el símbolo no existe o bien no se encuentra. En tal caso, para evitar que aparezca un mensaje de error, lo que hacemos es declarar **OnError GoTo**, de modo que cuando falle, directamente salte al párrafo Fallo, llame al método **DetenerSistema** y finalice.
    b. Que no se produzcan errores pero la fuente esté vacía. En tal caso también llamamos a **DetenerSistema** y terminamos.
    c. Que no se produzca errores y la fuenta tenga datos. En tal caso, continuamos con el proceso.
7. Si el proceso continúa, entonces podemos crear el objeto indicador pues ya tenemos una fuente base. Los indicadores se generan mediante el método de la clase ClaseDS **NewIndicator**. Este método siempre requiere de una fuente base, por lo que cada vez que queramos trabajar con indicadores, sabemos que previamente tendremos que crear su fuente correspondiente. El método NewIndicator nos pide el código del indicador (en nuestro ejemplo "RSI"), el objeto fuente y los parámetros del indicador. Como no vamos a modificar las bandas inferior y superior del RSI, las definimos directamente.
8. Si el indicador contiene datos, entonces procedemos a la descarga de datos. En otro caso, llamamos a **DetenerSistema** y acabamos.

Ya sólo falta definir la parte correspondiente a la descarga de datos. En este procedimiento, también usaremos variables locales que nos facilitarán la comprensión del proceso.

Las variables a usar serán las siguientes:

Bar: Esta variable la usaremos como contador de barras. Como nos vamos a recorrer el histórico completo de la fuente, la posición de barra n la guardaremos en Bar.
RSI: En esta variable guardaremos el valor devuelto por el indicador para la posición indicada en Bar.
RSIAnt: Como vamos a calcular la tendencia dada por el RSI, necesitaremos el valor anterior del RSI para ver la dirección que toma. En RSIAnt iremos guardando el último valor del RSI que vayamos calculando.
DifPrecios: En esta variable guardaremos la proporción entre apertura y cierre.

Fila: En esta variable iremos almacenando el número de fila que toca a la hora de escribir en excel. Conforme vayamos escribiendo, aumentaremos el valor de fila, que inicialmente valdrá 8 pues es la primera fila libre.
x: La variable x la usaremos como un valor méramente informativo. Como el proceso de descarga puede durar algunos segundos, vamos a indicar en cada momento cuanto nos falta para acabar calculando la proporción porcentual entre Bar y el tamaño total de barras. El resultado lo guardamos en x, y éste valor lo mostraremos en pantalla mediante el método de Excel Application.StatusBar.

El código del procedimiento **IniciarDescarga** quedará de la siguiente manera:

```
Public Sub IniciarDescarga()
Dim Bar As Long
Dim RSI As Double
Dim RSIAnt As Double
Dim DifPrecios As Double
Dim fila As Long
Dim x As Double
On Error GoTo Fallo
   fila = 8
   RSIAnt = 2147483647 'valor nulo
   For Bar = 1 To Fuente.Size
      'Info de descarga
      x = Int((Bar / Fuente.Size) * 100)
      Application.StatusBar = x & "% descargado"
      'fecha
      Range("A" & fila) = Fuente.GetBarValues(Bar).Date
      'apertura
      Range("B" & fila) = Round(Fuente.GetBarValues(Bar).Open, 4)
      'cierre
      Range("C" & fila) = Round(Fuente.GetBarValues(Bar).Close, 4)
      'diferencia %
      DifPrecios = Round((Range("C" & fila) - Range("B" & fila)) / Range("B" & fila), 3)
      Range("D" & fila) = DifPrecios
      Range("D" & fila).NumberFormat = "0.00%"
      If DifPrecios > 0 Then Range("D" & fila).Font.Color = RGB(0, 255, 0)
      If DifPrecios < 0 Then Range("D" & fila).Font.Color = RGB(255, 0, 0)
      'barras a partir de las cuales existe RSI
      If Bar >= Range("B5").Value Then
         'valor RSI
         RSI = Round(RSIData.Value(Bar, 1), 2)
         If RSI <> 2147483647 Then
            Range("E" & fila) = RSI
            'control de tendencia
            If RSIAnt <> 2147483647 Then
               If RSIAnt <= 30 And RSI > 30 Then TendenciaRSI = 1
               If RSIAnt >= 70 And RSI < 70 Then TendenciaRSI = -1
               If TendenciaRSI = 1 Then
                  Range("F" & fila) = "Alcista"
                  Range("F" & fila).Font.Color = RGB(0, 255, 0)
               ElseIf TendenciaRSI = -1 Then
                  Range("F" & fila) = "Bajista"
                  Range("F" & fila).Font.Color = RGB(255, 0, 0)
               End If
            End If
            RSIAnt = Range("E" & fila).Value
         End If
      End If
      fila = fila + 1
```

```
    Next Bar
    '---
    DetenerSistema
Fallo:
    If Err.Number <> 0 Then DetenerSistema
End Sub
```

En el procedimiento, hemos hecho lo siguiente:

1. RSIAnt lo inicializamos a ValorNulo para que la primera vez que tengamos un valor del RSI no calcule la tendencia, puesto que aún no tenemos dos valores para comparar.
2. Recorremos el histórico de la fuente mediante la declaración For To… Next. La variable Bar marcará la posición n de cada momento. El valor inicial será 1 y el último valor, es decir, la última barra descargada del gráfico se obtiene mediante la propiedad Size del objeto Fuente.
3. Para cada posición, escribimos en la celda correspondiente la fecha, apertura y cierre de la barra. Para ello, usamos el método GetBarValues del objeto Fuente.
4. Calculamos también la diferencia entre apertura y cierre. Para darle un valor añadido al dato, le aplicamos el formato porcentual usando la propiedad NumberFormat y además pintamos el valor de rojo o azul dependiendo de su la diferencia es positiva o negativa usando la propiedad Font.Color.
5. Las primeras barras del gráfico no tendrán valores del indicador RSI, por eso hay que esperar a que el valor de Bar sea igual o superior al periodo del indicador. Llegado a este punto, escribimos el valor del RSI.
6. Por último, calculamos la tendencia comparando el valor del RSI con el del RSIAnt y la posición de estos dos valores respecto a la banda inferior o superior. Aplicamos las reglas de tendencia descritas al principio y según sea en cada caso, almacenamos el valor correspondiente en la variable TendenciaRSI.
7. Una vez se alcanza el final del histórico, el programa sale del bucle For y por último, llamamos al método DetenerSistema para liberar los objetos y finalizar.

Hecho esto, tenemos terminado el proyecto y por tanto podemos cerrar el editor de VisualBasic.

Para finalizar desactivaremos el **Modo diseño** desde el menú **Programador** de Excel.

4º Visualizar la información

Antes de nada es necesario confirmar que en Visual Chart V estamos conectados al servidor de datos, y por otra parte, y que en nuestra licencia tenemos permisos para recibir información en tiempo real del activo que queramos probar.

A modo de ejemplo vamos a descargar datos del activo de BBVA.

El código completo en Visual Chart para este valor es 010060BBVA.MC. Es importante tener en cuenta que hay que indicar el código completo de cualquier producto que deseemos usar.

Por norma general, el código de cada producto seguirá la siguiente nomenclatura:

010 + código del mercado + símbolo

Si no indicamos el código completo, Visual Chart no puede encontrar el valor, y por lo tanto, no enviará ningún dato a Excel.

Una vez hemos introducido el código en la celda B4, rellenamos el resto de parámetros de entrada, o bien dejamos los que aparecen por defecto.

Por último, pulsamos **DESCARGAR,** y en este momento empezará el proceso de descarga de datos.

Cuando termine la descarga, el programa automáticamente destruirá el vínculo entre Visual Chart y Excel (puesto que sabemos que se habrá llamado al método DetenerSistema), y podremos tranquilamente cerrar nuestra hoja Excel.

COMTraderInterfaces

INTRODUCCIÓN

El servidor **COMTraderInterfaces** proporciona acceso a la operativa de Visual Chart V. Para su funcionamiento, es necesario que el programa esté ejecutándose y conectado a los servidores de datos e intermediación.

Como vemos en la imagen, este servidor actúa como intermediario entre VisualChart y la aplicación cliente (por ejemplo, Microsoft Excel).

El cliente hace peticiones a **COMTraderInterfaces** y éste notifica los resultados a través de eventos. De la misma forma, también podrá solicitar en cualquier momento algunos de los datos que se hayan recibido previamente mediante el uso de métodos.

Para poder hacer uso del servidor **COMTraderInterfaces**, la aplicación cliente debe de tener incorporada a su lista de referencias la clase **VisualChart Trader Library 1.0**

Esta librería incluye una serie de objetos, eventos y métodos que se detallan en las siguientes páginas.

OBJETOS

Los objetos de la librería **COMTraderInterfaces** son los que se indican a continuación:

- VCT_Order
- VCT_QueryFilter
- VCT_OpenPosition
- VCT_CloseOperation
- VCT_Balance
- VCT_Account
- VCT_MarketOrdersInfo
- VCT_Trader

VCT_Order. Este objeto encapsula la información relevante de una orden, tanto para enviarla al mercado como para recibirla actualizada en los distintos eventos del servidor.

Las propiedades más relevantes de cara al trabajo del usuario (por ejemplo para el envío de una orden) serían las siguientes:

- **Account.** Indica la cuenta con la que se quiere operar (un mismo usuario puede disponer de varias cuentas).
- **SymbolCode.** Símbolo sobre el que se lanza la orden (010060TEF.MC para Telefónica, por ejemplo).
- **OrderType.** Tipo de orden que se quiere lanzar.
- **OrderSide.** Especifica si se trata de una compra o una venta.

- **Volume.** Volumen que se quiere negociar en la orden.
- **Price y StopPrice.** Habrá que rellenar uno o los dos en función del tipo de orden de que se trate.
- **VolumeRestriction.** Aquí se indica la restricción de volumen que se quiere aplicar sobre la orden.
    - Si contiene la restricción VCT_VR_Hide (volumen oculto), es necesario rellenar el dato HideVolume indicando el volumen oculto.
    - Si contiene la restricción VCT_VR_MinVolume (volumen mínimo), es necesario rellenar el dato MinVolume indicando el volumen mínimo.
- **TimeRestriction.** Aquí se indica la restricción de tiempo que se quiere aplicar sobre la orden.
    - Si contiene la restricción VCT_TR_Date (hasta una fecha) es necesario rellenar el dato ValidDate con la fecha de validez deseada.
- **OrderID (identificador de orden).** Se trata de otra propiedad interesante que se puede consultar una vez que se ha enviado la orden. Será necesario utilizarlo, por ejemplo, en las modificaciones y cancelaciones de una orden ya enviada.

Algunos tipos de órdenes como OCO, OSO y Bracket, generan varias relacionadas entre sí.

Las propiedades **RelatedId** y **RelatedType** permiten determinar la relación existente entre ellas:

- En los casos de OSO y Bracket, en los que existe una orden principal, la orden tiene como RelatedId su propio identificador.

- En el resto de casos, RelatedId es el identificador de la orden con la que tiene la relación indicada en RelatedType.

Propiedad	Descripción
Account As String	Nombre de la cuenta.
SymbolCode As String	Código del valor.
OrderType As enumVCTOrderType	Tipo de orden.
OrderSide As enumVCTOrderSide	Indica si se trata de una compra o una venta.
Volume As Double	Volumen de la orden.
Price As Double	Precio de la orden.
StopPrice As Double	Precio de disparo de la orden.
ValidDate As Date	Fecha de validez.
UserOrderID As String	Identificador del usuario. No se usa internamente (el usuario puede usarlo para asociar un identificador propio a la orden).
OrderID As String	Identificador de la orden.
UserName As String	Usuario.
Date As Date	Fecha.
RemainVolume As Double	Volumen no ejecutado todavía.
Error As String	Último error (si lo hay).
Status As enumVCTOrderStatus	Estado de la orden.
Location As enumVCTOrderLocation	Localización de la orden.
Source As enumVCTSource	Origen desde el que se lanzó la orden.
Currency As String	Moneda
AvgPrice As Double	Precio medio de ejecución de la orden
VolumeRestriction As enumVCTVolumeRestriction	Restricción de volumen de la orden.

TimeRestriction As enumVCTTimeRestriction	Restricción de tiempo de la orden.
HideVolume As Long	Volumen oculto.
MinVolume As Long	Volumen mínimo.
RelatedId As String	Identificador de la orden asociada.
RelatedType As enumVCTRelatedType	Tipo de relación de la orden con su orden asociada (si la tuviese).

VCT_QueryFilter. Contiene la información del filtro a aplicar a la consulta de órdenes (ver GetOrders de VCT_Order).

Propiedad	Descripción
RowQueryLimit As Long	Limita el número de resultados de la consulta.
Filter As enumVCTOrderFilter	Tipo de órdenes que se desea consultar.
Account As String	Nombre de la cuenta.
StartDate As Date	Fecha inicial de la consulta.
EndDate As Date	Fecha final de la consulta.

VCT_OpenPosition. Contiene información sobre la posición abierta.

Propiedad	Descripción
Account As String	Nombre de la cuenta.
SymbolCode As String	Código del valor.
Description As String	Descripción.
Side As enumVCTOrderSide	Tipo de orden (compra o venta).
Volume As Double	Volumen de la posición abierta.
Currency As String	Moneda.
Price As Double	Precio.
Profit As Double	Ganancia.
Date As Date	Fecha.
PositionStatus As enumVCTOpenPositionStatus	Estado de la orden.

VCT_ClosedOperation. Contiene información sobre las operaciones cerradas.

Propiedad	Descripción
Account As String	Nombre de la cuenta.
SymbolCode As String	Código del valor.
Volume As Double	Volumen.
Currency As String	Moneda.
BoughtPrice As Double	Precio de compra.
SoldPrice As Double	Precio de venta.
Profit As Double	Beneficio.
BoughtDate As Date	Fecha de compra.
SoldDate As Date	Fecha de venta.

VCT_Balance. Contiene información sobre el saldo y capital disponible para operar.

Propiedad	Descripción
PortfolioValue As Double	Cartera.
NetWorth As Double	Patrimonio.
Margins As Double	Garantías.
OutstandingBalance As Double	Poder de compra.
Cash As Double	Efectivo.
MarginCash As Double	Efectivo retenido.
CashByLeverage As Double	Disponible por apalancamiento.

VCT_Account. Contiene información sobre la cuenta con la que se opera.

Propiedad	Descripción
Account As String	Nombre de la cuenta.
Description As String	Descripción de la cuenta.
Balance As VCT_Balance	Balance (ver VCT_Balance).

VCT_MarketOrdersInfo. Contiene información sobre un mercado.

Propiedad	Descripción
MarketCode As String	Código del mercado.

VCT_Trader.Este objeto representa la interfaz principal del servidor COM y es el punto de partida tanto para el envío de órdenes como para la recepción de eventos relacionados con las mismas.

Propiedad	Descripción
Accounts As VCT_Accounts	Devuelve una colección con todas las cuentas del usuario.
MarketOrdersInfos As VCT_MarketOrdersInfos	Devuelve una colección de VCT_MarketOrdersInfo con información sobre los tipos de órdenes y las restricciones soportadas por los distintos mercados.
ClosedOperations As VCT_ClosedOperations	Devuelve una colección con las operaciones cerradas.
OpenPositions As VCT_OpenPositions	Devuelve una colección con las posiciones abiertas.
Orders As VCT_Orders	Devuelve una colección con las órdenes (todas, activas…) existentes en este momento en Visual Chart

EVENTOS

OnOrderInMarket(Order As VCT_Order). Indica que la orden está en mercado.
OnModifiedOrder(Order As VCT_Order). La orden ha sido modificada.
OnCancelledOrder(Order As VCT_Order). La orden ha sido cancelada.
OnTotalExecutedOrder(Order As VCT_Order). Se ha ejecutado completamente la orden.
OnError(Order As VCT_Order). Ha ocurrido un error relacionado con la orden indicada. (consultar la propiedad Error de VCT_Order)
OnPartialExecutedOrder(Order As VCT_Order). Se ha ejecutado parcialmente la orden.
OnNewOrderLocation(Order As VCT_Order). La orden ha cambiado de localización. (ver enumVCTOrderLocation)
OnChangedOpenPositions(Account As String). Han cambiado las posiciones abiertas de la cuenta indicada.
OnNewClosedOperations(Account As String). Hay nuevas operaciones cerradas de la cuenta indicada.
OnChangedBalance(Account As String). El balance de la cuenta indicada ha cambiado.
OnServerShutDown(). Este evento se ejecuta justo antes de que el servidor deje de estar disponible. Es útil para notificar a otras aplicaciones que ya no pueden usar el servidor o realizar tareas propias del cierre de la aplicación.

MÉTODOS

A continuación se detallan los métodos que se pueden utilizar con los objetos de la librería **ComTraderInterfaces.**

VCT_MarketOrdersInfo

GetSupportedOrders(OrderType() As enumVCTOrderType). Devuelve un array con todos los tipos de órdenes soportados por el mercado.
GetVolumeRestriction(VolumeRestriction() As enumVCTVolumeRestriction). Devuelve un array con todos los tipos de restricciones de volumen soportadas por el mercado.
GetTimeRestriction(TimeRestriction() As enumVCTTimeRestriction). Devuelve un array con todos los tipos de restricciones de tiempo soportadas por el mercado.

VCT_Trader

SendOrder(Account As String, SymbolCode As String, OrderType As enumVCTOrderType, OrderSide As enumVCTOrderSide, Volume As Double, Price As Double, StopPrice As Double, VolumeRestriction As enumVCTVolumeRestriction, TimeRestriction As enumVCTTimeRestriction,[HideVolume As Object], [MinVolume As Object], [ValidDate As Object]) As String. Ejecuta una nueva orden con los valores indicados en los parámetros. Devuelve el identificador de la orden generada.
SendOrderEx(Order As VCT_Order). Ejecuta una nueva orden con los valores indicados en el parámetro.
ModifyOrder(OrderId As String, Volume As Double, Price As Double, StopPrice As Double, [HideVolume As Object], [ValidDate As Object]). Modifica la orden cuyo identificador sea OrderId asignándole los valores indicados en los parámetros.
CancelOrder(OrderId As String). Cancela la orden indicada.
SendOCOOrder(Order1 As VCT_Order, Order2 As VCT_Order) As String(). Envía dos órdenes enlazadas entre sí mediante una relación OCO. Devuelve los identificadores de las dos órdenes.
SendOSOOrder(MainOrder As VCT_Order, OrderToSend As VCT_Order) As String(). Envía dos órdenes enlazadas entre sí mediante una relación OSO y devuelve el identificador de las dos órdenes generadas. MainOrder es la orden que se envía inmediatamente mientras que OrderToSend es la orden

que se enviará cuando se ejecute MainOrder.
SendBracketOrder(MainOrder As VCT_Order, LimitOrder As VCT_Order, StopOrder As VCT_Order) As String(). Envía una orden Bracket (envía una orden de entrada a mercado junto con una limitada de beneficios y un stop de pérdidas) y devuelve el identificador de las tres órdenes generadas.
GetOrders(QueryFilter As VCT_QueryFilter) As VCT_Orders. Devuelve una colección con las órdenes que verifique el filtro QueryFilter. Estos resultados los devuelve el servidor por lo que se puede consultar cualquier información aún no estando disponible en Visual Chart.

COLECCIONES

Todas las colecciones (excepto **VCT_Orders)** que tiene algunos métodos extra, tienen la misma interfaz y el mismo funcionamiento. **Se usan para devolver un conjunto de datos del mismo tipo.**

La siguiente tabla muestra las colecciones existentes y el tipo de objeto que contienen:

Colección	Tipo de objeto
VCT_Orders	VCT_Order
VCT_OpenPositions	VCT_OpenPosition
VCT_ClosedOperations	VCT_ClosedOperation
VCT_Balances	VCT_Balance
VCT_Accounts	VCT_Account
VCT_MarketOrdersInfos	VCT_MarketOrdersInfo

Propiedad	Descripción
Count As Long	Devuelve el número de elementos de la colección.
Item(index As Object) As XXX	Devuelve un ítem de la colección (XXX se refiere al tipo de la colección mostrado en la segunda columna de la tabla anterior)
_NewEnum	Todas las colecciones implementan este método oculto para permitir el recorrido por los distintos ítems de la colección usando foreach.

VCT_Orders. Esta colección, además de los métodos ya citados, contiene otros que permiten obtener colecciones de órdenes de un tipo determinado (ejecutadas, activas o canceladas).

Active() As VCT_Orders. Devuelve una colección con las órdenes activas que contiene la colección.
Executed() As VCT_Orders. Devuelve una colección con las órdenes ejecutadas que contiene la colección.
Cancelled() As VCT_Orders. Devuelve una colección con las órdenes canceladas que contiene la colección.

EJEMPLO PRÁCTICO DEL USO DEL SERVIDOR COMTraderInterfaces

En el ejemplo que veremos a continuación, vamos a crear un panel de comandos que nos permita comprar o vender sobre un valor determinado.

Este panel, además de tener los comandos de compra y venta, nos permitirá elegir entre 3 tipos de órdenes distintas:

- Entrar a mercado **por lo mejor**
- Enviar una orden a mercado en **stop (por lo mejor)**
- Enviar una orden a mercado **limitada**

También debe permitir elegir el volumen de contratos y el precio de entrada (siempre que no sea por lo mejor)

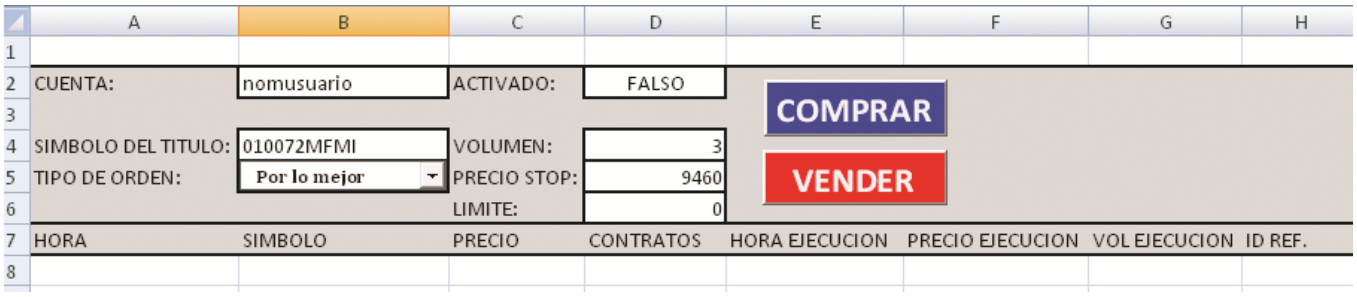
Una vez más vamos a usar como aplicación cliente Microsoft Excel. Aprovecharemos el libro Trading Tools con el que venimos trabajando en este manual, y usaremos la hoja 3 de dicho libro, ya que en la hoja 1 y en la hoja 2 tenemos los ejemplos previos.

*Es necesario tener en cuenta que puesto que vamos a hacer uso del objeto **COMTraderInterface** es necesario haber añadido a la lista de referencias la librería VisualChart Trader Library 1.0*

Los pasos a seguir se pueden resumir en los siguientes:

1º Preparar el escenario

Lo primero que haremos será diseñar la interfaz visual para introducir y mostrar la información. La hoja 3 de nuestro libro Excel debe quedar de la siguiente manera:



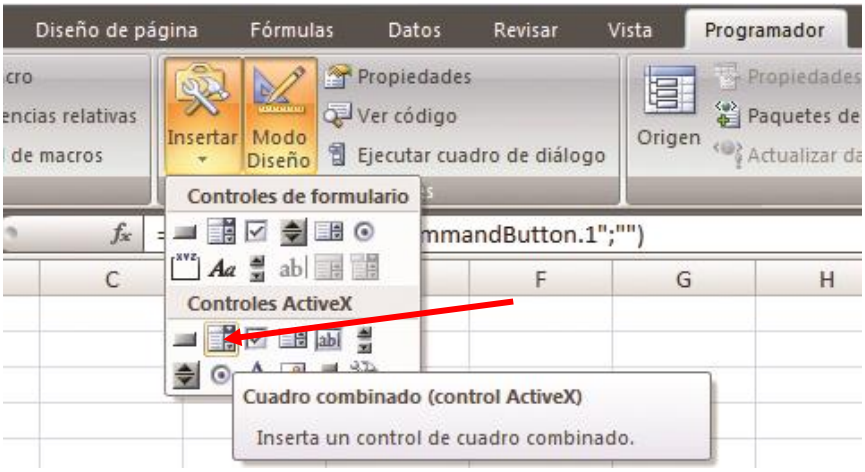
Del escenario diseñado, cabe destacar los tres objetos ActiveX que hemos añadido:

- Un cuadro combinado
- Dos botones de comando (COMPRAR/VENDER)

El cuadro combinado lo vamos a usar para que el usuario elija entre los distintos tipos de órdenes que pueden ejecutarse. A través de VisualChart V, siempre y cuando su broker se lo permita, se puede utilizar una amplia gama de tipos de órdenes. Para este ejemplo vamos a usar tres tipos:

- Ordenes a mercado por lo mejor
- Ordenes limitadas.
- Ordenes en stop por lo mejor.

Para crear el cuadro combinado, accedemos desde el menú **Programador** a la opción **Insertar → Cuadro Combinado (active X).**



Insertamos este objeto justo en la celda B5 para que quede alineado con el resto de celdas.

Hecho esto, accedemos al panel de propiedades del objeto (accionando el botón derecho del ratón sobre el objeto) y modificamos los siguientes campos:

Name	CboTipoOrdenes
BorderStyle	1-fmBorderStyleSingle
Height	15,75
Width	105

La lista de valores del cuadro combinado se rellena desde el código de programación.

Para ello, vamos a crear un procedimiento en el código que se llame CargaTipoOrdenes y que quedaría de la siguiente manera:

```
Private Sub CargaTipoOrdenes()
    CboTipoOrdenes.AddItem "Por lo mejor"
    CboTipoOrdenes.AddItem "Limitada"
    CboTipoOrdenes.AddItem "Stop Por lo Mejor"
    CboTipoOrdenes.Value = "Por lo mejor"
End Sub
```

Otra opción que tiene el usuario, en lugar de añadir un objeto de tipo cuadro combinado, es usar el mismo método que se llevó a cabo en el ejemplo del servidor VCDataSource. En dicho ejemplo, veíamos cómo activar la opción de validación de datos. En tal caso, habría que hacer esto mismo para la celda B5, generando una lista en celdas auxiliares con los tres textos indicados en el procedimiento CargaTipoOrdenes.

Los botones **COMPRAR** y **VENDER** son similares a los botones **ACEPTAR** y **PARAR** de la hoja ejemplo de tiempo real.

Recordar que que para crear controles en una hoja Excel, es necesario seleccionar el menú de **Programador**, activar la opción **Modo Diseño**, y posteriormente seleccionar la opción **Insertar → Botón de comando (active X).**
Como hicimos con los otros dos botones del ejemplo anterior, accedemos al panel de propiedades del objeto y desde éste modificamos los parámetros correspondientes al nombre:

Para el botón COMPRAR

Name	BtnComprar
Caption	COMPRAR
BackColor	&H00FF0000&

Para el botón VENDER

Name	BtnVender
Caption	VENDER
BackColor	&H000000FF&

A continuación, pulsamos dos veces sobre cada botón para que se generen los eventos **Click** en el editor de Visual Basic.

Recordemos que este tipo de evento nos indica que cada vez que el usuario pulse sobre el botón, se llevará a cabo lo que indiquemos dentro del evento.

2º Programación de procedimientos

En primer lugar vamos a crear un objeto llamado **ClaseTrader** que sea del tipo **VCT_Trader.**

```
Public WithEvents ClaseTrader As COMTraderInterfacesLib.VCT_Trader
```

A diferencia de los otros dos ejemplos que hemos visto, en este caso el proceso de trabajo no es lineal, es decir, no se trata de que el usuario active con un botón y se inicie la descarga de información, ya que en esta ocasión, se trata es de que el usuario pueda pulsar los botones de compra o venta aleatoriamente según sus necesidades.

Es por esto que tenemos que buscar un punto de inicio externo a los botones de compra y venta.

El funcionamiento va a ser el siguiente:

1. Inicializar el escenario cada vez que se abra el libro Excel (mediante el evento **Workbook_Open**)
2. Generar compras o ventas según el botón que pulse el usuario y según la información de las celdas.
3. Guardar un backup de las operaciones realizadas en las celdas contiguas.
4. Liberar el objeto **VCT_Trader** una vez se cierre el libro Excel (mediante el evento **Workbook_BeforeClose**).

Para el paso 1 vamos a crear un procedimiento que llamaremos **IniciarEscenario** y que quedará de la siguiente manera:

```
Public Sub IniciarEscenario()
   Range("A8:H1000").ClearContents
   Range("D2") = False
   LiberarObjetoTrader
   CargaTipoOrdenes
End Sub
```

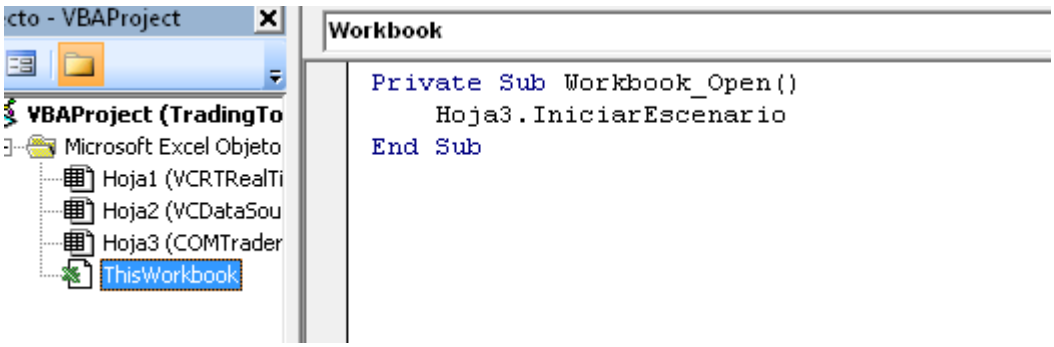
Lo que hemos hecho es lo siguiente:

1. Limpiar el contenido de las celdas desde la fila 8.
2. La celda D2 la inicializamos a FALSO. Esta celda nos va a servir para saber si el objeto **VCT_Trader** ha sido creado o no.
   - Si la celda indica FALSO, quiere decir que no está creado y que habrá que crearlo.
   - Si la celda indica VERDADERO, ya está creado y no hay que hacer nada al respecto.
3. Liberamos el objeto **ClaseTrader** como medida de seguridad.

4. Rellenamos el combo **Tipo de Ordenes** en caso de que lo estemos usando.

El procedimiento **IniciarEscanario** lo llamaremos cuando se produzca el evento **Worbook_Open**. Este evento debemos declararlo desde el objeto **ThisWorbook**.

Para abrir el código de este objeto, desde la ventana del Explorador de Proyectos seleccionamos el objeto ThisWorkbook y pulsamos sobre él dos veces:



Desde el código de **ThisWorkBook** definimos el procedimiento **Workbook_Open** e indicamos que queremos hacer una llamada al procedimento **IniciarEscenario** perteneciente a la Hoja3 del libro.

Si observamos, el procedimiento **IniciarEscenario** lo hemos declarado como público. Esto permite que se pueda acceder a dicho procedimiento desde un elmento externo.

Para el paso 2 de nuestro desarrollo, vamos a rellenar los códigos de **BtnComprar_Click** y **BtnVender_Click** que previamente hemos debido generar.

Ambos botones harán una llamada a un mismo procedimiento que llamaremos **GenerarOrden**.

Como la rutina para el envio de órdenes es igual, tanto en el caso de enviar compras como en el caso de enviar ventas, recurrimos a este procedimiento externo que sencillamente solicitará el signo de la orden para diferenciar entre ventas y compras.

El código de **GenerarOrden** quedará de la siguiente forma:

```
Private Sub GenerarOrden(SignoOrden As enumVCTOrderSide)
Dim Cuenta As String
Dim Simbolo As String
Dim TipoOrden As enumVCTOrderType
Dim Limite As Double
Dim PrecioSTOP As Double
Dim Volumen As Integer
Dim NewID As String
    Cuenta = Range("B2").Text
    Simbolo = Range("B4").Text
    TipoOrden = SelectOrderT(CboTipoOrdenes.Text)
    Volumen = Range("D4").Value
    PrecioSTOP = Range("D5").Value
    Limite = Range("D6").Value
    'Guardar informacion previa
    Range("A" & UlFila) = Time
    Range("B" & UlFila) = Simbolo
    Range("C" & UlFila) = PrecioSTOP
    If TipoOrden = VCT_OT_BestStop Then Range("C" & UlFila) = PrecioSTOP
    Range("D" & UlFila) = Volumen
    '-- Enviar orden
```

```
    NewID = ClaseTrader.SendOrder(Cuenta, Simbolo, TipoOrden, SignoOrden, Volumen, Limite,
PrecioSTOP)
    Range("G" & UlFila) = 0
    Range("H" & UlFila) = NewID
    UlFila = UlFila + 1
  End Sub
```

Como vemos, se solicita como parámetro de entrada el valor **SignoOrden**, que es de tipo **enumVCTOrderSide**. Esta clase de enumeración forma parte de las enumeraciones de la clase **COMTraderInterfaces** y distingue entre órdenes de compra (**VCT_OS_Buy**) y órdenes de venta (**VCT_OS_Sell**).

En cuanto al contenido del código, hemos hecho lo siguiente:

1. Definir los distintos campos obligatorios para lanzar una orden, tales como cuenta, símbolo, volumen… Extraemos el valor de estos campos de las distintas celdas que hemos declarado en nuestra interfaz.
2. Cabe destacar que el campo **TipoOrden** es una variable de tipo **enumVCTOrderType**, también de la familia de la clase **COMTraderInterfaces**. Más adelante comentaremos cómo hemos rellenado dicha variable.
3. Mostramos en pantalla la información inicial al envio de la orden, tal y como la hora de envío, el símbolo o el precio. Además de esta información, mostraremos en pantallas los datos relativos a la orden una vez se ha ejecutado. Para ello nos apoyaremos en los eventos propios del objeto **VCT_Trader**.
4. Enviamos la orden usando el método del objeto **VCT_Trader SendOrder**. Además de los campos que hemos definido en este objeto, que son los campos obligatorios, el usuario puede también añadir información adicional a la orden, tal y como la fecha de validación, el volumen oculto, volumen mínimo, etc.
5. Por último, guardamos en la celda correspondiente el identificador ID de la orden, pues nos servirá posteriormente, y aumentamos el contador de filas UlFila. Esta variable es una variable global que debemos declarar al inicio del código y que iremos aumentando conforme añadamos nuevas órdenes.

Como hemos dicho anteriormente, la variable **TipoOrden** es una variable especial que debe ser rellenada con valores especiales.

Lo que haremos es asociar cada uno de estos valores con los posibles valores que el usuario puede seleccionar en nuestro cuadro combinado **CboTipoOrdenes**. Para hacer esto, añadimos en el código una función que, a diferencia de los procedimientos, devuelve un valor de salida. A esta función la llamaremos **SelectOrderT** y que quedará de la siguiente forma:

```
Private Function SelectOrderT(NameT As String) As enumVCTOrderType
Dim AUXValue As enumVCTOrderType
    Select Case NameT
        Case "Por lo mejor"
            AUXValue = VCT_OT_Best
        Case "Limitada"
            AUXValue = VCT_OT_Limit
        Case Else
            AUXValue = VCT_OT_BestStop
    End Select
    SelectOrderT = AUXValue
End Function
```

La función recibe la variable **NameT** (que será el valor seleccionado del cuadro combinado), y según sea su valor, devolverá el tipo de orden correspondiente en función de la enumeración **enumVCTOrderType**.

Ya tenemos definido el procedimiento al que llamaremos desde los eventos **BtComprar_Click** y **BtnVender_Click**. Por tanto, podemos seguidamente definir ambos eventos:

```
Private Sub BtnComprar_Click()
   If Range("D2") = False Then
      Set ClaseTrader = New COMTraderInterfacesLib.VCT_Trader
      Range("D2") = True
      UlFila = 8
   End If
   GenerarOrden (VCT_OS_Buy)
End Sub
```

```
Private Sub BtnVender_Click()
   If Range("D2") = False Then
      Set ClaseTrader = New COMTraderInterfacesLib.VCT_Trader
      Range("D2") = True
      UlFila = 8
   End If
   GenerarOrden (VCT_OS_Sell)
End Sub
```

Como vemos, según sea el botón que pulsemos, el procedimiento **GenerarOrden** recibirá como valor de entrada **VCT_OS_Buy** o **VCT_OS_Sell**.

Además de la llamada a **GenerarOrden**, en ambos casos se realiza un chequeo para verificar si existe o no el objeto **VCT_Trader**. Esto lo sabemos gracias a la celda D2. Si la celda está en FALSO, el objeto no se ha declarado y hay que declararlo mediante la llamada **Set = New**.

Hecho una vez, no será necesario hacerlo nuevamente, ya que seguiremos usando el objeto tantas veces como el usuario considere oportuno. También aprovechamos el chequeo para inicializar la variable **UlFila**.

El paso 3 consistirá en rellenar la información que falta de cada orden. Como hemos visto, antes de enviar una orden mediante el método **SendOrder**, escribimos en pantalla:

- Fecha
- Precio
- Tipo de orden que vamos a enviar a mercado

Hecho esto, aumentamos en 1 la variable **UlFila** que es la que controla la última fila escrita. Esto nos sirve para llevar un seguimiento de las órdenes enviadas.

Adicionalmente a la información guardada en el envío de órdenes, vamos a guardara también un registro de las órdenes ejecutadas. Para ello, nos vamos a valer del evento **OnTotalExecuteOrder** del objeto **VCT_Order** y, a modo de soporte, del evento **OnPartialExecuteOrder**.

El evento **OnTotalExecuteOrder** se activará cada vez que se ejecute una orden de manera total en mercado. Si el volumen de contratos ejecutados en mercado es inferior al volumen total, entonces se activará el evento **OnPartialExecuteOrder**, de modo que ya sólo se activará el evento **OnTotal** una vez se ejecuten los contratos que queden pendientes.

Ejemplo. Si lanzamos una orden en stop de compra de 4 contratos, puede pasar lo siguiente:

1. Que se ejecuten los 4 contratos al únisono → Se activará el evento OnTotal con volumen = 4.
2. Que se ejecuten menos de 4 (supongamos 3) → Se activará primero el evento OnPartial con volumen = 3, y luego se activará el evento OnTotal con volumen = 1 (el contrato restante).

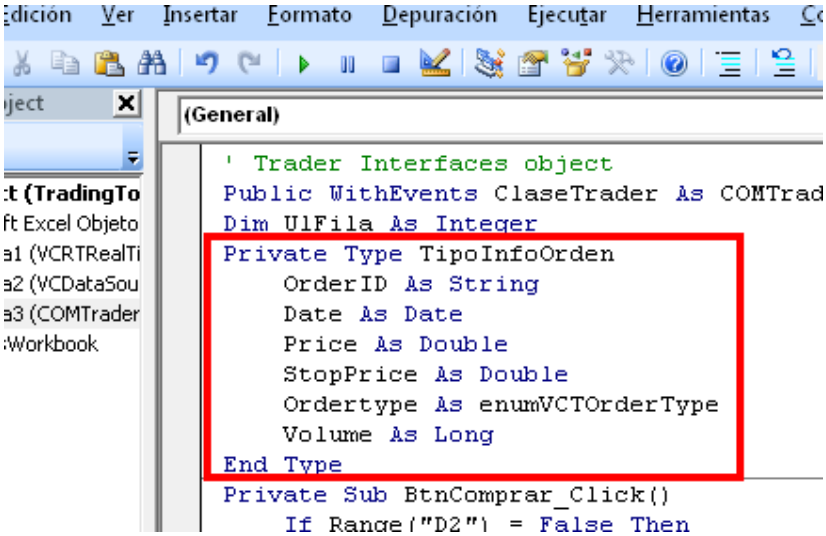
Lógicamente, puede pasar que no llegue a activarse el evento OnTotal si permanece pendiente de ejecución algún contrato.

Como la estructura a realizar en uno u otro evento es la misma, volveremos a hacer uso de un procedimiento externo al que llamaremos desde cada evento.

A este procedimiento lo vamos a denominar **OrdenEjecutada** y recibirá como parámetro de entrada la información procedente de la orden ejecutada parcial o totalmente, según sea el caso.

En lugar de pasarle como parámetros de entrada varios campos con la información de la orden ejecutada, vamos a definir un tipo de variable propia que tenga como argumentos cada uno de los datos que queremos pasarle al procedimiento; de este modo, sólo tendremos que declarar como parámetro de entrada una sola variable que sea de esta clase nuestra.

A la nueva clase de variable la vamos a llamar **TipoInfoOrden**, la declaramos al principio del código y quedará de la siguiente forma:



Hecho esto, se puede escribir el código del procedimiento **OrdenEjecutada** el cual quedará así:

```
Private Sub OrdenEjecutada(Order As TipoInfoOrden)
Dim Salir As Boolean: Dim i As Integer
Dim Encontrada As Boolean
   i = 8: Salir = False: Encontrada = False
   While Not Salir
      If Range("H" & i) = "" Then
         Salir = True
      Else
         If Order.OrderID = Range("H" & i) Then
            Salir = True
            Encontrada = True
         Else
            i = i + 1
         End If
      End If
   Wend
   If Encontrada Then
      Range("E" & i) = DatePart("h", Order.Date) & ":" & DatePart("n", Order.Date) & ":" & DatePart("s",
Order.Date)
      Range("F" & i) = Order.Price
      If Order.Ordertype = VCT_OT_BestStop Then Range("F" & i) = Order.StopPrice
      Range("G" & i) = Range("G" & i) + Order.Volume
   End If
End Sub
```

Tal y como hemos comentado, el procedimiento recibe la variable **Order** de tipo **TipoInfoOrden** que habrá recibido del evento **OnTotal** o del evento **OnPartial**.

Lo que hacemos en OrdenEjecutada es lo siguiente:

1. Nos recorremos las distintas filas donde puede haber información en busca de la fila cuya columna H coincida con el valor ID de la variable **Order**.

2. Una vez encontremos la fila correspondiente (Encontrada = Verdadero), escribimos en las columnas correspondientes los campos contenidos en la variable **Order**.

Existen dos puntos a destacar:

1. Las órdenes enviadas y ejecutadas pueden llevar dos tipos de precio distintos según sea el tipo de orden.

    - Si la orden es limitada, el precio de la orden está referenciado en el campo Price.
    - Si la orden es en stop, el precio de la orden está referenciado en el campo StopPrice. Las órdenes a mercado no llevan ningún precio asociado.

2. En la columna G vamos guardando el volumen de contratos ejecutados. Puestoque en caso de que se ejecuten contratos parciales, podemos acceder varias veces al procedimiento **OrdenEjecutada** con la misma orden, vamos aumentando el contenido de la celda con cada nueva llamada al procedimiento.

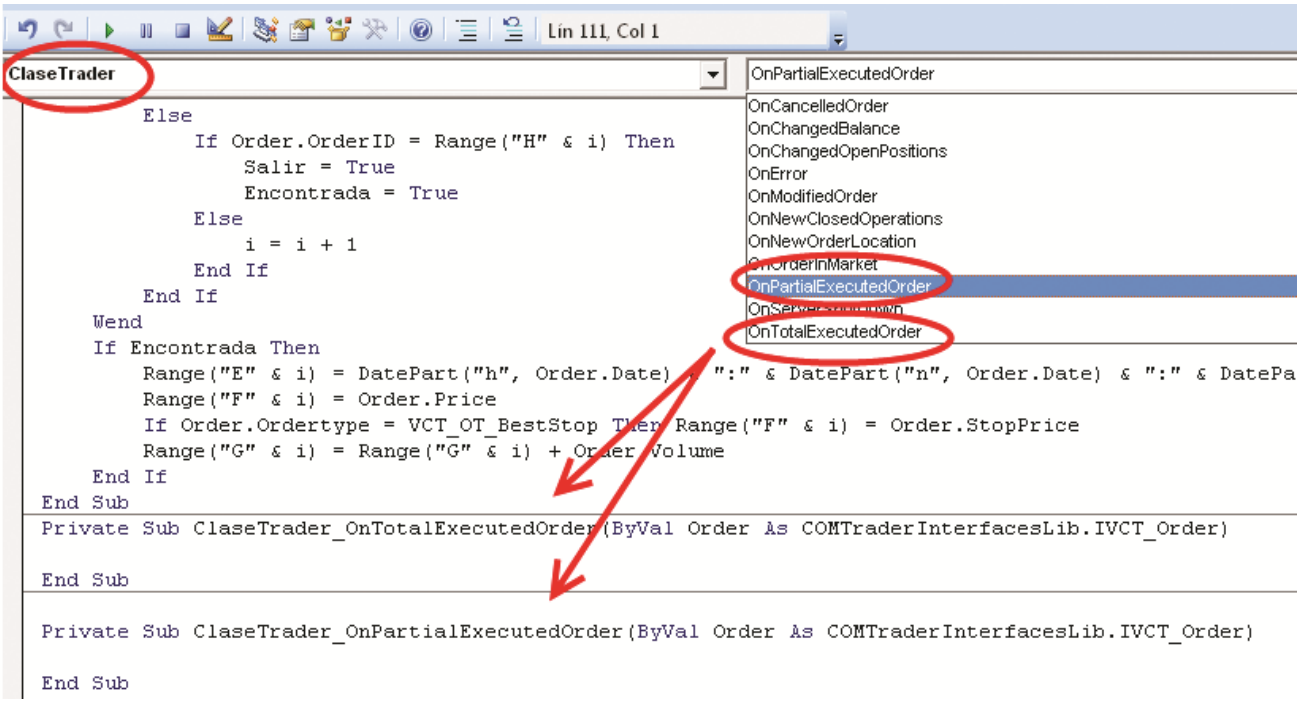
```
            Else
                i = i + 1
            End If
        End If
    Wend
    If Encontrada Then
        Range("E" & i) = DatePart("h", Order.Date) & ":" & DatePart("n", Order.Date)
        Range("F" & i) = Order.Price
        If Order.Ordertype = VCT_OT_BestStop Then Range("F" & i) = Order.StopPrice
        Range("G" & i) = Range("G" & i) + Order.Volume
    End If
End Sub
'-- EVENTOS
```

Definido el procedimiento **OrdenEjecutada**, el siguiente paso será declarar los siguientes eventos:

- OnTotalExecuteOrder
- OnPartialExecuteOrder

Para ello, en la pestaña de selección de objetos, escogemos el objeto **ClaseTrader,** y automáticamente, en la pestaña de procedimientos, aparecerán los propios de este objeto.

Del listado de procedimientos, seleccionamos el evento **OnTotalExecuteOrder** y a continuación **OnPartialExecuteOrder**.



Una vez que los seleccionemos, se generarán los eventos en el código. Ahora sólo falta rellenar cada uno de los eventos que, como hemos dicho antes, serán exactamente iguales:

```
'-- EVENTOS
Private Sub ClaseTrader_OnTotalExecutedOrder(ByVal Order As COMTraderInterfacesLib.IVCT_Order)
Dim NewOrder As TipoInfoOrden
   NewOrder.OrderID = Order.OrderID
   NewOrder.Date = Order.Date
   NewOrder.Price = Order.Price
   NewOrder.StopPrice = Order.StopPrice
   NewOrder.Ordertype = Order.Ordertype
   NewOrder.Volume = Order.Volume
   OrdenEjecutada NewOrder
End Sub
```

```
Private Sub ClaseTrader_OnPartialExecutedOrder(ByVal Order As COMTraderInterfacesLib.IVCT_Order)
Dim NewOrder As TipoInfoOrden
   NewOrder.OrderID = Order.OrderID
   NewOrder.Date = Order.Date
   NewOrder.Price = Order.Price
   NewOrder.StopPrice = Order.StopPrice
   NewOrder.Ordertype = Order.Ordertype
   NewOrder.Volume = Order.Volume
   OrdenEjecutada NewOrder
End Sub
```

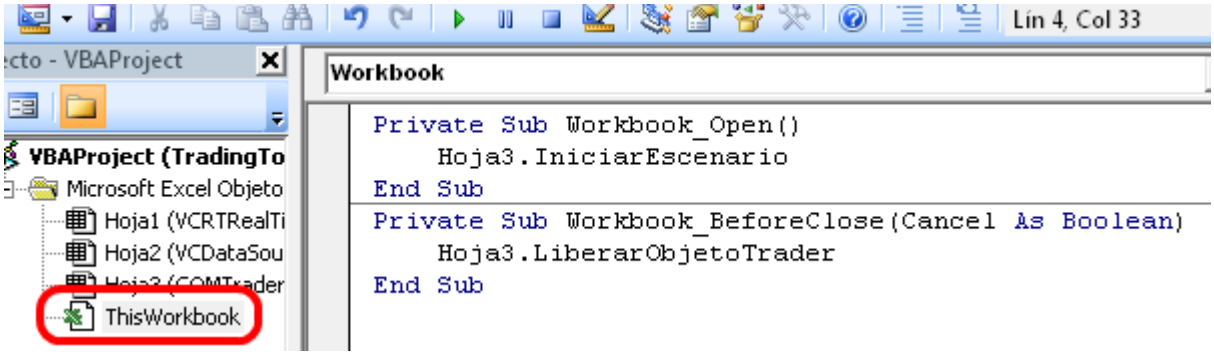
Sencillamente lo que hacemos es crear una variable de tipo **TipoInfoOrden** a la que llamaremos **NewOrder**, rellenarla con los datos correspondientes de la orden ejecutada que nos ha llegado, y por último, llamar al procedimiento **OrdenEjecutada** pasándole como parámetro la variable **NewOrder**.

Hecho esto, únicamente queda liberar el objeto **VCT_Trader** (paso4). Para ello, vamos a crear un nuevo procedimiento al que llamaremos **LiberarObjetoTrader** y que quedará de la siguiente forma:

```
Public Sub LiberarObjetoTrader()
   Set ClaseTrader = Nothing
End Sub
```

Puesto que el funcionamiento de nuestro desarrollo no es cíclico, no podremos liberarlo hasta que el usuario decida dejar de usar el libro excel, es decir, una vez que lo cierre. Para ello, usaremos el evento **Workbook_BeforeClose** el cual se activa justo antes de cerrar el libro.

Al igual que hicimos para el paso 1, debemos declararlo desde el objeto **ThisWorbook**. Para abrir el código de este objeto, desde la ventana del Explorador de Proyectos seleccionamos el objeto ThisWorkbook y pulsamos sobre él dos veces.



Desde el código de ThisWorkBook definimos el procedimiento Workbook_BeforeClose e indicamos que queremos hacer una llamada al procedimento **LiberarObjetoTrader** perteneciente a la Hoja3 del libro.

Una vez más, observamos que el procedimiento LiberarObjetoTrader ha sido declarado como de dominio público, lo que nos permite hacer la llamada a dicho procedimiento desde ThosWorbook.

Con esto queda por finalizado el código, por lo tanto ya podemos guardar el libro y cerrarlo.

3º Ejecución

Al abrir la hoja de Excel se pone en marcha el evento Worbook_Open que es el que nos marca el inicio del desarrollo.

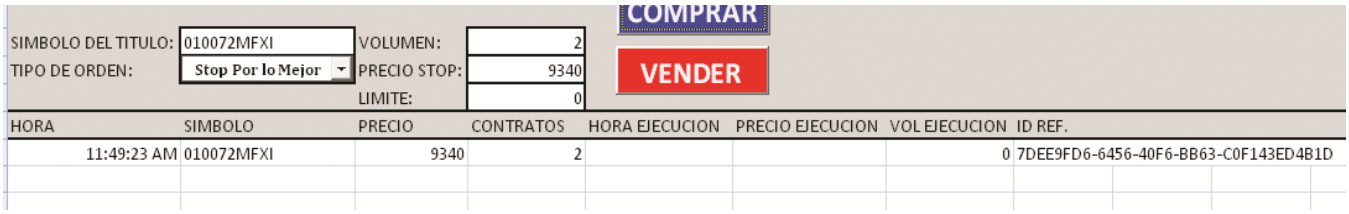
Lo siguiente será confirmar que tenemos **Visual Chart V** abierto, que estamos conectados al servidor de intermediación (modo real o simulado).

Anotamos la cuenta de usuario que vayamos a usar en la celda de **CUENTA** (B2).

Rellenamos el resto de celdas según nos interese. Por ejemplo, vamos a probar a enviar una orden en stop sobre el futuro de IBEX continuo de dos contratos.

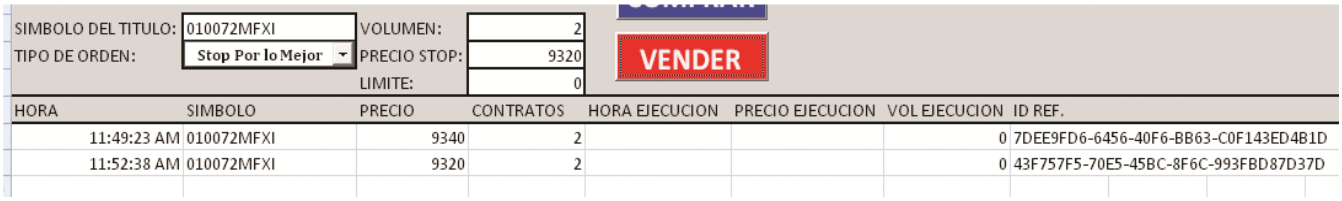
Para ello, rellenamos las celdas de la siguiente manera:

Pulsamos el botón **COMPRAR** y vemos que al instante se rellena la primera fila de datos:

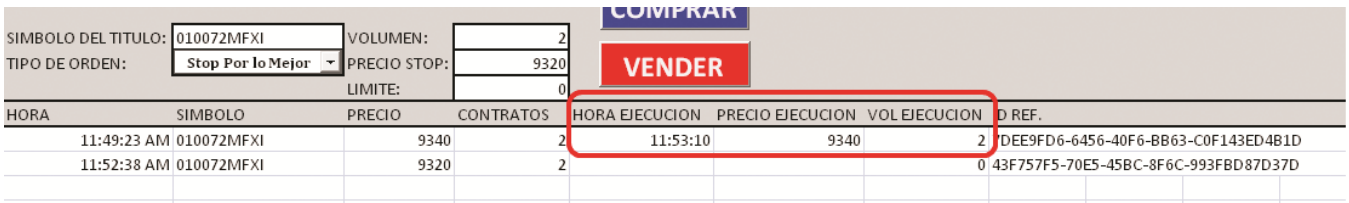


La orden permanece activa en mercado y por tanto no se ejecuta. Es por esto que las celdas de orden ejecutada permanecen vacías.

A continuación enviamos la orden contraria por si el mercado se da la vuelta. Lanzaremos una orden en stop de venta. Para ello, modificamos la celda **PRECIO STOP = 9320** y pulsamos **VENDER**. De esta forma se rellenará la siguiente fila de datos:



Finalmente, la orden de COMPRA termina ejecutándose, por lo que automáticamente se rellena la fila correspondiente a dicha orden:



Por último, cerramos el libro excel de modo que liberamos el objeto VCT_Trader, rompiendo los vínculos con VisualChart.

***Importante**: En el mercado permanece una orden abierta de compra y otra activa de venta independientemente de que hayamos finalizado el uso de la hoja excel.*

*El usuario deberá tener esto en cuenta. No obstante, existe la posiblidad de cancelar órdenes usando el propio servidor **COMTraderInterfaces**, por lo que si el usuario lo requiere, puede añadir a su código el método correspondiente a dicha acción.*

VCContributor

INTRODUCCIÓN

Una de las nuevas funcionalidades de Visual Chart V es el manejo de contribuciones. Las contribuciones son una herramienta mediante la cual los usuarios pueden compartir con otros usuarios, recibir o enviar, cualquiera de los siguientes elementos:

- Noticias
- Alertas
- Propuestas de órdenes
- Páginas completas de un espacio de trabajo (gráficos, tablas, páginas Web…)

Esta nueva herramienta también puede ser utilizada a través de los servidores COM mediante el servidor VCContributor.

Como se ve en la imagen, el servidor actúa como intermediario entre Visual Chart y la aplicación cliente (por ejemplo Microsoft Excel).

Para poder hacer uso de éste servidor, la aplicación cliente debe de tener incorporada a su lista de referencias la clase **VisualChart Contributor library 1.0.**

Al igual que ocurre con el resto de librerías, dispone de una colección de objetos, eventos y métodos que podemos usar para la manipulación de datos.

Para poder enviar contribuciones, el servidor se vale de **métodos**. Mientras que para recibir las noticias enviadas por otros usuarios o por canales de contribución, el servidor se vale de **eventos**.

OBJETOS

Los objetos de la librería **VCContributor** son los siguientes:

- VCC_Filters
- VCC_Device
- VCC_Devices
- VCC_IAlert
- VCC_ITrader
- VCC_IGraphic
- VCC_ILiveMsg
- VCC_Ilive

VCC_Filters. Representa una colección de filtros. Estos son clasificaciones de las distintas contribuciones que pueden enviarse desde un canal y son propios de cada uno.La configuración de filtros de un canal, permite a Visual Chart decidir qué contribuciones deja pasar y cuales se filtran (las que no interese recibir en un momento dado).

Propiedad	Descripción
Count As Long	Número de filtros del canal.
Item(Index As Long) As String	Valor del filtro.

VCC_Device. Visual Chart dispone de distintos dispositivos donde se encapsulan o muestran distintas funcionalidades de las contribuciones. Un dispositivo gráfico, por ejemplo, representa una página del espacio de trabajo, mientras que un dispositivo de alertas se usa para mostrar alertas en Visual Chart.

- Cuando se envían contribuciones, los dispositivos que no estén activos no se envían junto con la contribución.
- Cuando se recibe una contribución, los dispositivos desactivados indican que no se incluyeron al enviar la contribución y, por tanto, no tienen valores relevantes.

Propiedad	Descripción
Active As Bool	Activa o desactiva el dispositivo.

VCC_Devices. Representa una colección de dispositivos. La utiliza **VCC_ILiveMsg** para configurar dispositivos extra antes del envío de una contribución. También contiene, cuando se reciben contribuciones, dichos dispositivos.

Propiedad	Descripción
Count As Long	Número de dispositivos de la colección.
Item(Index As Long) As VCC_Device	Dispositivo de la colección.

VCC_IAlert. Implementa la interfaz de **VCC_Device** y representa una alerta de Visual Chart. Las alertas que más prioridad tengan se mostrarán antes que las que tienen menor prioridad.

Propiedad	Descripción
Text As String	Texto/mensaje de la alerta.
Priority As enumVCCAlertPriority	Prioridad de la alerta.
LifeTime As Long	Tiempo de vida de la alerta.
Sound As String	Sonido asociado a la alerta.

VCC_ITrader. Este objeto implementa la interfaz **VCC_Device** y representa una orden (ver VCT_Order). La propiedad Order contiene una propuesta de orden que puede ejecutarse directamente o previa modificación de alguna de sus propiedades.

Propiedad	Descripción
Order As VCT_Order	Orden asociada al dispositivo VCC_ITrader. Es la orden que se propone al usuario de VisualChart.

VCC_IGraphic. Este objeto implementa la interfaz VCC_Device y representa una página del espacio de trabajo que el usuario puede visualizar en Visual Chart.

Propiedad	Descripción
WksXML As String	XML con la serialización de la página del espacio de trabajo. Si

	en lugar de un XML se indica el nombre de una página del espaci de trabajo, VisualChart usa el XML asociado a dicha página.

VCC_ILiveMsg. VCC_ILiveMsg contiene una descripción completa de una contribución. Se utiliza, tanto para configurar la contribución que se desea enviar, como para consultar la información de una contribución recibida.

Propiedad	Descripción
Channel As String	Canal de la contribución
Users As String	Usuarios que recibirán la contribución. Puede estar vacía si la contribución se envía solo al canal o contener uno o varios nombres d usuario (separados por " , " o " ; ")
SaveHistoric As Bool	Vale **True** si la contribución se guarda en el histórico. Las contribuciones que se guardan en el histórico pueden consultarse desde VisualChart después de ser enviadas. Si no se guardan en el histórico solo se reciben cuando han sido enviadas.
Header As String	Título.
Topic As String	Tema.
Body As String	Mensaje.
iGraphic As VCC_IGraphic	Página del espacio de trabajo asociada a la contribución (si Active es **True**).
iTrader As VCC_ITrader	Orden propuesta asociada a la contribución (si Active es **True**).
iAlert As VCC_IAlert	Alerta asociada a la contribución (si Active es **True**).
ExtraDevices As VCC_Devices	Dispositivos adicionales (más órdenes propuestas, por ejemplo) asociados a la contribución.
Filters As VCC_Filters	Filtros de la contribución.
RelatedCode As String	Código del símbolo asociado a la contribución.
CtbType As Long	Tipo de contribución (ver enumVCCCtbType)

VCC_ILive Este objeto representa la interfaz principal de la librería VCContributor. Permite enviar contribuciones y da acceso a las contribuciones que se reciben por los canales previamente registrados.

EVENTOS

A continuación se detallan los distintos eventos que se pueden producir.

VCC_ILive

OnNewILiveMsg(iLiveMsg As VCC_ILiveMsg). Este evento se ejecuta cada vez que se recibe una contribución. En iLiveMsg está toda la información de dicha contribución.
OnServerShutDown (). Este evento se ejecuta justo antes de que el servidor deje de estar disponible. Es útil para notificar a otras aplicaciones que ya no pueden usar el servidor o realizar tareas propias del cierre de la aplicación.

MÉTODOS

En esta página se detallan los distintos métodos que se utilizan con la librería **VCContributor**.

VCC_Devices

Add(Device As VCC_Device). Agrega un dispositivo a la colección.
Remove(Index As Long). Elimina un dispositivo de la colección.
Clear(). Elimina todos los dispositivos de la colección.

VCC_ILiveMsg

SendMsg(ILive As VCC_ILive). Envía la contribución configurada en VCC_ILiveMsg usando el objeto ILive. Dicho objeto debe haberse declarado previamente y será el encargado de enviar la contribución.

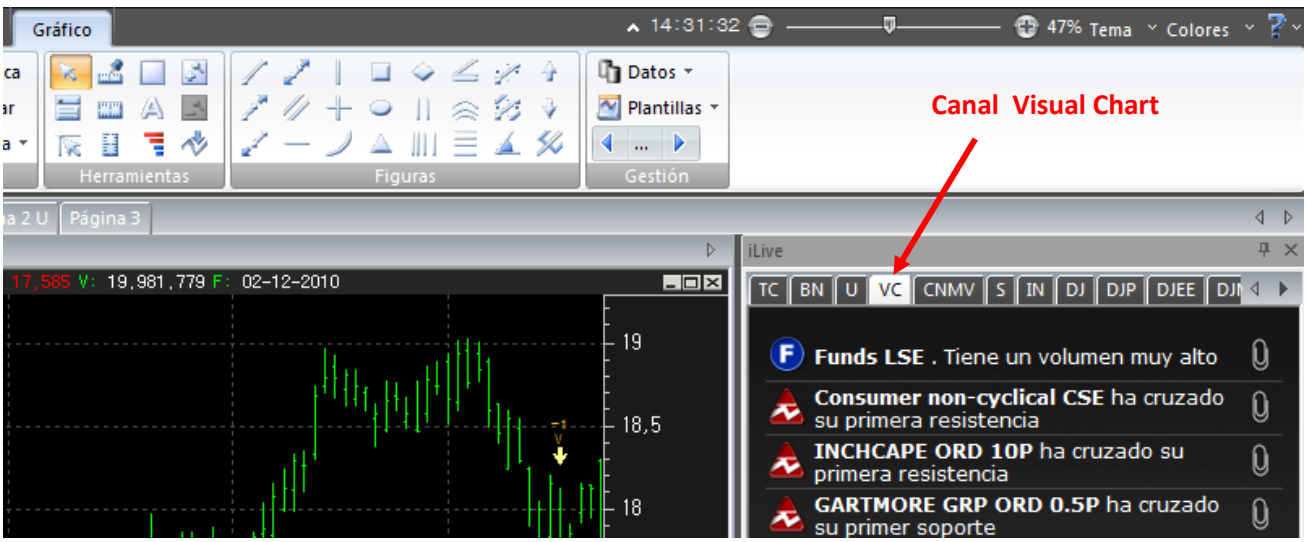
VCC_ILive

SendMsg(iLiveMsg As VCC_ILiveMsg). Envía una contribución con los valores especificados en el parámetro iLiveMsg.
RequestChannel(Channel As String). Registra el canal especificado (Channel es el nombre del canal). El registro permite recibir contribuciones de dicho canal. Por cada registro que se haga debe llamarse después a CancelChannel para liberar recursos.
CancelChannel(Channel As String). Libera los recursos asociados al registro de un canal y, si ya no quedan registros de dicho canal, dejan de recibirse contribuciones del mismo.

EJEMPLO PRÁCTICO DEL USO DEL SERVIDOR VCContributor

Como se ha indicado anteriormente, la recepción de contribuciones supone una de las nuevas herramientas disponibles en Visual Chart V.

Existen distintos canales de noticias que el usuario puede recibir, dependiendo del acceso que tenga a uno u más de ellos. Entre estos canales de noticias, existe uno de libre acceso para todos los usuarios de tiempo real y delay llamado **VisualChart (VC)**.



En nuestro ejemplo, vamos a filtrar las noticias que nos lleguen desde este canal, interesándonos sólo por las que contengan información sobre soportes y resistencias. Una vez más vamos a hacer uso de nuestro libro Excel **Trading Tools,** utilizando en este caso la hoja 4.

Es necesario tener en cuenta que se debe haber añadido a la lista de referencias la librería VisualChart Contributor 1.0.

Los pasos a seguir se pueden resumir en los siguientes:

1º Preparación del escenario

Como para el resto de ejemplo, lo primero que haremos será diseñar la interfaz visual para introducir y mostrar la información. La hoja de Excel debe quedar así:

Como vemos, incorpora 2 objetos del tipo botón de comando (ACEPTAR y PARAR), similares a los utilizados en los ejemplos anteriores.

Recordar que para crear controles en una hoja excel es necesario seleccionar el menú de **Programador**, activar la opción **Modo Diseño**, y posteriormente seleccionar la opción **Insertar → Botón de comando (active X).**

Una vez que se han insertado los botones, accedemos al panel de propiedades de cada uno de los objetos y modificamos los siguientes campos:

Para el botón ACTIVAR

Name	BtnActivar
Caption	ACTIVAR
BackColor	&H00FF0000&

Para el botón PARAR

Name	BtnParar
Caption	PARAR
BackColor	&H000000FF&

A continuación, pulsamos dos veces sobre cada botón para que se generen los eventos **Click** en el editor de Visual Basic.

Recordemos que este tipo de evento nos indica que cada vez que el usuario pulse sobre el botón, se llevará a cabo lo que indiquemos dentro de dicho evento.

2º Programación de procedimientos

Una vez definido el escenario, vamos a crear un objeto llamado **ClaseVCILive** que sea del tipo **VCC_ILive.**

```
Public WithEvents ClaseVCILive As VCC_ILive
```

El funcionamiento de nuestro desarrollo va a seguir los siguientes pasos:

1. Iniciar el programa una vez el usuario pulse el botón **ACTIVAR**. Inicializamos el escenario y creamos el objeto VCC_ILive.
2. Recibir las contribuciones que nos lleguen y filtrarlas según lo establecido anteriormente.
3. Una vez hemos filtrado las noticias, mostrar en pantalla aquellas que pasen el filtro.
4. Liberar el objeto VCC_ILive una vez el usuario pulse el botón **PARAR**.

Para el primer paso, vamos a rellenar el evento **BtnActivar_Click** que quedará de la siguiente manera:

```
Private Sub BtnActivar_Click()

   Range("A7:G6000").ClearContents
   DetenerSistema
   BtnActivar.Enabled = False
   BtnParar.Enabled = True
   Set ClaseVCILive = New VCC_ILive
   NuevaFila = 7
End Sub
```

Lo que hemos hecho es lo siguiente:

1. Limpiar el contenido de las celdas desde la fila 7.
2. Llamar al procedimiento **DetenerSistema**. Este procedimiento sencillamente libera al objeto VCC_ILive. Hacemos esta llamada como medida de seguridad.
3. Activar el botón **PARAR** y desactivar el botón **ACTIVAR**. De este modo tenemos una percepción visual de cuándo está el programa en funcionamiento.
4. Crear el objeto **ClaseVCILive** y reiniciar la variable **NuevaFila** a 7.

La variable NuevaFila marcará la siguiente fila que escribiremos en pantalla y que iremos aumentando conforme lleguen nuevos mensajes. Esta variable la declararemos al principio del código.

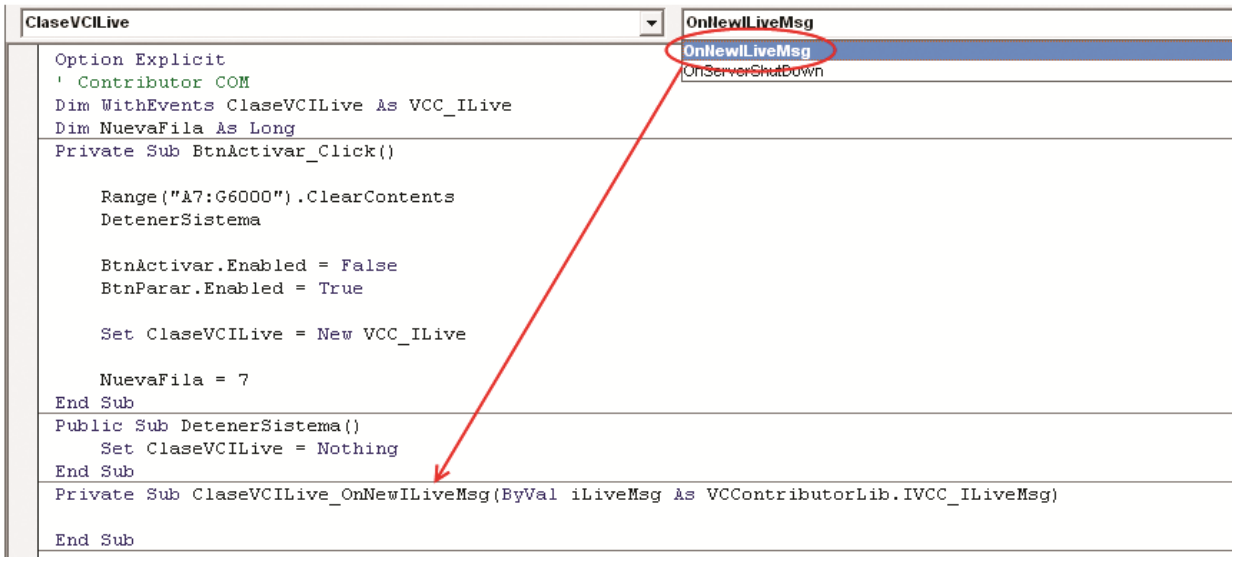
```
Public WithEvents ClaseVCILive As VCC_ILive

Dim NuevaFila As Long
```

El procedimiento **DetenerSistema** quedará de la siguiente forma:

```
Public Sub DetenerSistema()
   Set ClaseVCILive = Nothing
End Sub
```

Para el paso 2 de nuestro desarrollo, vamos a necesitar declarar el evento **OnNewILiveMsg** del objeto **ClaseVCILive**. Cada vez que se reciba una nueva contribución desde cualquier canal que el usuario tenga activado, este evento se activará.

Para declarar el evento, en la pestaña de selección de objetos, escogemos el objeto **ClaseVCILive,** y automáticamente, en la pestaña de procedimientos, aparecerán los propios de este objeto. Del listado de procedimientos, seleccionamos el evento OnNewILiveMsg.



Una vez que los seleccionemos, se generará el evento en el código, y sólo quedaría rellenarlo.

Antes de esto, vamos a explicar cómo filtrar los mensajes que nos llegan de Visual Chart. Como decíamos al principio, sólo vamos a registrar los mensajes que sean del tipo **Soportes y Resistencias**.

Una de las propiedades de las contribuciones, consiste en que podemos definir si la contribución enviada está relacionada con fundamentales, patrones, análisis técnico, etc.

Las noticias que lleguen relacionadas con soportes y resistencias serán del tipo chartistas, por lo tanto filtraremos los mensajes por este tipo.

El objeto que recibimos con el evento **OnNewILiveMsg**, **iLiveMsg**, contiene una propiedad llamada **CtbType** que hace referencia al tipo de noticia. Esta propiedad puede tener cualquier clase de valor del tipo **enumVCCCtbType**.

Añadimos a continuaicón una función que traduzca cada uno de los valores posibles de la enumeración enumVCCtbType, a fin de conseguir una mejor comprensión de la información.

La función se llamará **DameNomTipo** y quedará de la siguiente forma:

```
Private Function DameNomTipo(ByVal value As VCContributorLib.enumVCCCtbType) As String
    If value = VCC_CT_None Then
        DameNomTipo = "Sin Tipo"
    ElseIf value = VCC_CT_Indicators Then
        DameNomTipo = "Indicadores"
    ElseIf value = VCC_CT_Charting Then
        DameNomTipo = "Chartistas"
    ElseIf value = VCC_CT_Patterns Then
        DameNomTipo = "Patrones velas"
    ElseIf value = VCC_CT_Outstanding Then
        DameNomTipo = "Eventos"
    ElseIf value = VCC_CT_BreakingNews Then
        DameNomTipo = "Destacados"
    ElseIf value = VCC_CT_Fundamentals Then
        DameNomTipo = "Fundamentales"
    ElseIf value = VCC_CT_Ratings Then
        DameNomTipo = "Ratios"
    End If
End Function
```

Cuando se active el evento y recibamos un nuevo objeto iLiveMsg, llamaremos a esta función para saber si el tipo de mensaje es de tipo Chartistas, que es precisamente lo que andamos buscando.

Aclarado esto, el contenido del evento **OnNewILiveMsg** quedará de la siguiente forma:

```
Private Sub ClaseVCILive_OnNewILiveMsg(ByVal iLiveMsg As VCContributorLib.IVCC_ILiveMsg)
Dim Canal As String
Dim Simbolo As String
Dim Tipo As enumVCCCtbType
Dim Titulo As String
Dim Tema As String
Dim Mensaje As String
    'Guardar info
    Canal = iLiveMsg.Channel
    Simbolo = iLiveMsg.RelatedCode
    Tipo = iLiveMsg.CtbType
    Titulo = iLiveMsg.Header
    Tema = iLiveMsg.Header
    Mensaje = iLiveMsg.Body
    'Filtrar mensajes
    If Canal = "Visual Chart" Then
        If DameNomTipo(iLiveMsg.CtbType) = "Chartistas" Then
            If iLiveMsg.Topic = "Cruce de resistencia" Or iLiveMsg.Topic = "Cruce de soporte" Then
                Cells(NuevaFila, 1) = Now
                Cells(NuevaFila, 2) = Canal
                Cells(NuevaFila, 3) = Simbolo
                Cells(NuevaFila, 4) = "Chartistas"
                Cells(NuevaFila, 5) = Titulo
                Cells(NuevaFila, 6) = Tema
                Cells(NuevaFila, 7) = Mensaje
                NuevaFila = NuevaFila + 1
            End If
        End If
    Else
        Range("A1") = "Nuevo mensaje a las: " & Time & ": " & Titulo
    End If
End Sub
```

Lo que hemos hecho es lo siguiente:

1. Para una mejor comprensión de la información, hemos ido guardando cada uno de los elementos del objeto iLiveMsg en variables según el tipo de elemento: Cana, símbolo, Tipo, etc.
2. Filtramos cada uno de los mensajes, de manera que sólo se guarden aquellos que cumplan el siguiente criterio:
    - Que el canal sea Visual Chart
    - Que el tipo de mensaje sea tipo Chartistas
    - Que el subtítlo de la noticia sea, o bien Cruce de resistencia, o bien Cruce de soporte.
3. Si se cumplen los criterios, entonces escribimos, en la fila correspondiente, cada uno de los elementos del mensaje.
4. A continuación, aumentamos el contador de filas. Esto sería el paso 3 de nuestro desarrollo.
5. Por último, como información adicional, cada vez que no se cumpla el criterio, vamos a escribir en la celda A1 la hora y el título del mensaje despreciado.Esto nos sirve para confirmar que estamos recibiendo las contribuciones, aunque como decimos, este dato es meramente informativo.

Finalizados los pasos 2 y 3, ya sólo nos faltaría realizar el paso 4, es decir, liberar el objeto VCC_ILive.

Para ello, rellenamos el contenido del evento **BtnParar_Click** de la siguiente forma:

```
Private Sub BtnParar_Click()
    BtnParar.Enabled = False
    BtnActivar.Enabled = True

    Range("A1").ClearContents

    DetenerSistema
End Sub
```

Sencillamente lo que hacemos es lo siguiente:

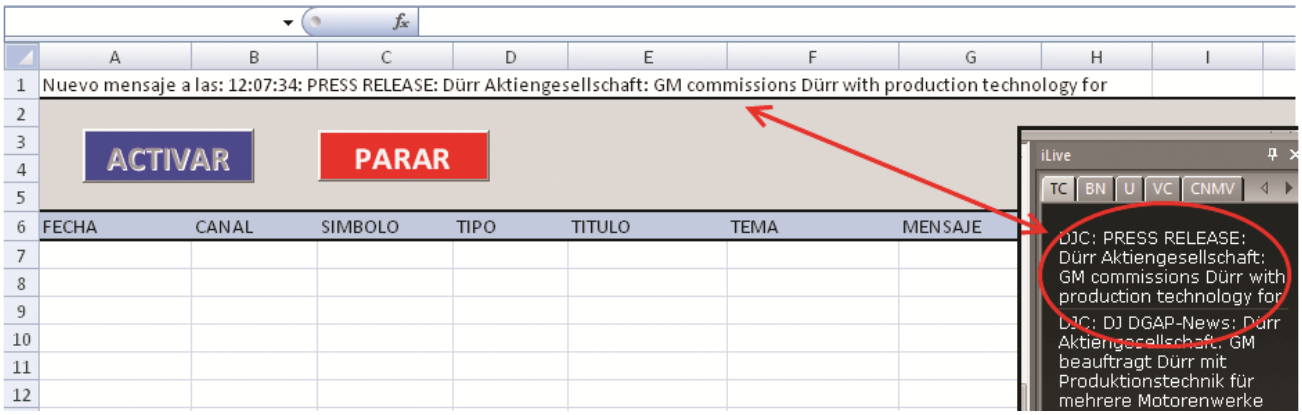
1. Volver a activar el botón ACTIVAR y a desactivar el botón PARAR (para tener una percepción visual de que hemos detenido el programa)
2. Limpiar la celda A1 de la información adicional
3. Llamar al procedimiento DetenerSistema para que libere al objeto VCC_ILive.

Con esto queda por finalizado el código, lo único que queda es desactivar el **Modo Diseño** de Excel.

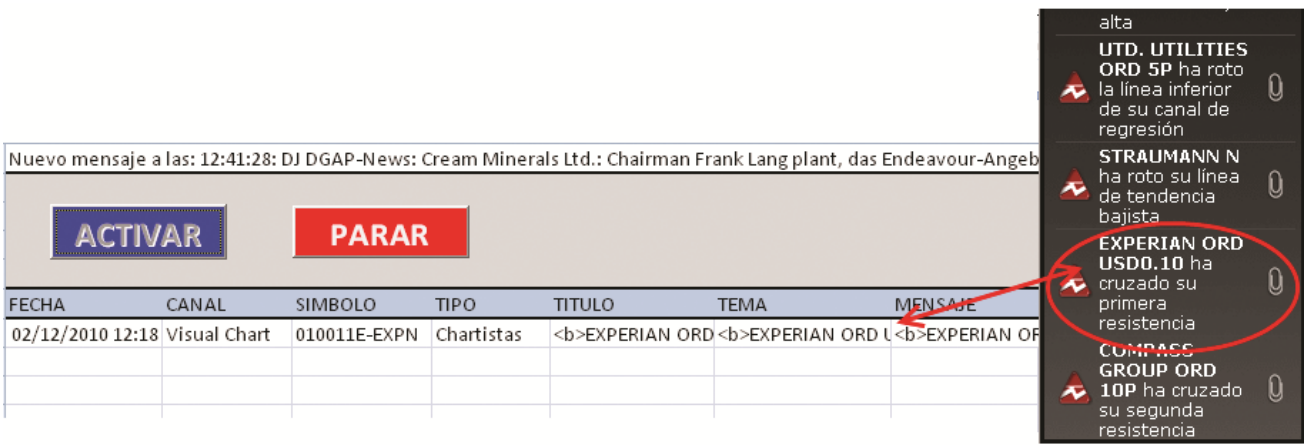
3º Visualizar la información

Antes de ejecutar la hoja para recibir la información, es necesario confirmar que tenemos conexión de tiempo real (delay) en Visual Chart.

A continuación, sólo tenemos que pulsar el botón ACTIVAR y veremos, que a medida que llegan mensajes, se van mostrando en la celda A1:



Cuando aparezca un mensaje que supere nuestro filtro, se guardará en la fila correspondiente:



APÉNDICE

ENUMERACIONES DE LAS DISTINTAS LIBRERÍAS

Las enumeraciones son los posibles valores que puede tomar un tipo de variable concreta. A continuación se facilita la descripción de éstas para cada una de las librerías.

ENUMERACIONES VCRealTime

Las variables propias de la librería **VCRealTime** son las siguientes:

EnumVCRTField. Contiene las definiciones de todos los campos que se pueden consultar o recibir.

Valor	Descripción
VCRT_Field_Code	Código del símbolo (Ej: 010060TEF.MC).
VCRT_Field_Last	Valor del último.
VCRT_Field_Evol	Evolución.
VCRT_Field_Buy1	Precio de compra en la primera posición.
VCRT_Field_Sale1	Precio de venta en la primera posición.
VCRT_Field_Time	Hora.
VCRT_Field_Diff	Diferencia.
VCRT_Field_Porcentual	Diferencia porcentual.
VCRT_Field_Los_Trades	Negocios.
VCRT_Field_Last_Vol	Valor del volumen último.
VCRT_Field_Volume	Volumen.
VCRT_Field_Vol_Mon	Volumen monetario.
VCRT_Field_Low	Mínimo.
VCRT_Field_High	Máximo.
VCRT_Field_Open	Apertura.
VCRT_Field_Bid_Size	Número de títulos ofertados en la compra.
VCRT_Field_Ask_Size	Número de títulos ofertados en la venta.
VCRT_Field_Previous	Precio Anterior.
VCRT_Field_Open_Int	Open Interés.
VCRT_Field_Average_P	Precio Medio.

VCRT_Field_Description	Descripción.
VCRT_Field_B_Ag	Agencia compradora.
VCRT_Field_S_Ag	Agencia vendedora.
VCRT_Field_Date	Fecha.
VCRT_Field_Value_Per_Point	Valor por punto.
VCRT_Field_Expiry_D	Fecha de expiración.
VCRT_Field_ISIN	Código ISIN.
VCRT_Field_Sub_Market	Submercado al que pertenece.
VCRT_Field_Minimum_Mov	Mínimo movimiento.
VCRT_Field_Decimals	Decimales.

EnumVCRTExtField. Contiene las definiciones de los campos extendidos que se pueden consultar o recibir. A continuación se detalla los distintos valores que puede tomar las variables de estos 2 tipos:

Valor	Descripción
VCRT_EF_FUND_Description_ES	Descripción de la empresa (español).
VCRT_EF_FUND_Description_EN	Descripción de la empresa (inglé).
VCRT_EF_FUND_Description_FR	Descripción de la empresa (francés).
VCRT_EF_FUND_Description_DE	Descripción de la empresa (alemán).
VCRT_EF_FUND_IndustryWeight	Ponderación del valor dentro de la industria.
VCRT_EF_FUND_SectorWeight	Ponderación del valor dentro del sector.
VCRT_EF_FUND_SubsectorWeight	Ponderación del valor dentro del subsector.
VCRT_EF_FUND_MarketWeight	Ponderación del valor dentro del mercado.
VCRT_EF_FUND_URL	URL de la we de la empresa.
VCRT_EF_FUND_MarketCap	Capitalización del mercado.
VCRT_EF_FUND_MarketCapDayInc	Aumento de la capitalización de mercado en el día.
VCRT_EF_FUND_Nempoyees	Número de empleados.
VCRT_EF_FUND_Noffices	Numero de oficinas/establecimientos.
VCRT_EF_FUND_Officers	Directivos de la empresa.
VCRT_EF_FUND_RefIndex	Indices de referencia a efectos comparativos (cáculo Beta).
VCRT_EF_FUND_SharesOut	Número de acciones.
VCRT_EF_FUND_MktSharesOut	Acciones en el mercado. Puede ser diferente del campo SharesOut si el símbolo cotiza en varios mercados.
VCRT_EF_FUND_FaceValue	Valor nominal de una acción.
VCRT_EF_FUND_NetIncome	Beneficio Neto.
VCRT_EF_FUND_NetIncome_B	Beneficio Neto Semestral.
VCRT_EF_FUND_NetIncome_Q	Beneficio Neto Trimetral.
VCRT_EF_FUND_PreTaxIncome	Beneficio neto antes de impuestos / BAI (Anual).
VCRT_EF_FUND_PreTaxIncome_B	Beneficio neto antes de impuestos / BAI (Semestral).
VCRT_EF_FUND_PreTaxIncome_Q	Beneficio neto antes de impuestos / BAI (Trimestral).
VCRT_EF_FUND_NetIncCont	Beneficio neto de actividades continuadas (Anual).
VCRT_EF_FUND_NetIncCont_B	Beneficio neto de actividades continuadas (Semestral).
VCRT_EF_FUND_NetIncCont_Q	Beneficio neto de actividades continuadas (Trimestral).
VCRT_EF_FUND_Sales	Ventas/Importe neto cifra negocio (Anual).

VCRT_EF_FUND_Sales_B	Ventas/Importe neto cifra negocio (Semestral).
VCRT_EF_FUND_Sales_Q	Ventas/Importe neto cifra negocio (Trimestral).
VCRT_EF_FUND_OpIncome	Beneficios Explotacion / Beneficios antes de impuestos e intereses (Anual).
VCRT_EF_FUND_OpIncome_B	Beneficios Explotacion / Beneficios antes de impuestos e intereses (Semestral).
VCRT_EF_FUND_OpIncome_Q	Beneficios Explotacion / Beneficios antes de impuestos e intereses (Trimestral).
VCRT_EF_FUND_OIBDA	Beneficio operativo antes de depreciación y amortización (Anual).
VCRT_EF_FUND_OIBDA_B	Beneficio operativo antes de depreciación y amortización (Semestral).
VCRT_EF_FUND_OIBDA_Q	Beneficio operativo antes de depreciación y amortización (Trimestral).
VCRT_EF_FUND_GrossIncome	Beneficio bruto (Anual).
VCRT_EF_FUND_GrossIncome_B	Beneficio bruto (Semestral).
VCRT_EF_FUND_GrossIncome_Q	Beneficio bruto (Trimestral).
VCRT_EF_FUND_ExtraIncome	Resultados extraordinarios/Resultados de actividades descontinuadas (Anual).
VCRT_EF_FUND_ExtraIncome_B	Resultados extraordinarios/Resultados de actividades descontinuadas (Semestral).
VCRT_EF_FUND_ExtraIncome_Q	Resultados extraordinarios/Resultados de actividades descontinuadas (Trimetral).
VCRT_EF_FUND_COGS	Coste de producción de las ventas (Anual).
VCRT_EF_FUND_COGS_B	Coste de producción de las ventas (Semestral).
VCRT_EF_FUND_COGS_Q	Coste de producción de las ventas (Trimetral).
VCRT_EF_FUND_OpExpenses	Costes operativos / Gastos de explotación (Anual).
VCRT_EF_FUND_OpExpenses_B	Costes operativos / Gastos de explotación (Semetral).
VCRT_EF_FUND_OpExpenses_Q	Costes operativos / Gastos de explotación (Trimestral).
VCRT_EF_FUND_TotalExpenses	Gastos totales (Anuales).
VCRT_EF_FUND_TotalExpenses_B	Gastos totales (Semestrales).
VCRT_EF_FUND_TotalExpenses_Q	Gastos totales (Trimestrales).
VCRT_EF_FUND_IncTaxes	Impuesto de sociedades (Anual).
VCRT_EF_FUND_IncTaxes_B	Impuesto de sociedades (Semestral).
VCRT_EF_FUND_IncTaxes_Q	Impuesto de sociedades (Trimestral).
VCRT_EF_FUND_MinorityInterests	Intereses minoritarios/Beneficios atribuibles a accionistas minoritarios (Anual).
VCRT_EF_FUND_MinortyInterests_B	Intereses minoritarios / Beneficios atribuibles a accionistas minoritarios (Semetral).
VCRT_EF_FUND_MinorityInterests_Q	Intereses minoritarios / Beneficios atribuibles a accionistas minoritarios (Trimetral).
VCRT_EF_FUND_NetNonOpLoss	Pérdidas no atribuibles a la explotación (Anual).
VCRT_EF_FUND_NetNonOpLoss_B	Pérdidas no atribuibles a la explotación (Semestral).
VCRT_EF_FUND_NetNonOpLoss_Q	Pérdidas no atribuibles a la explotación (Trimestral).
VCRT_EF_FUND_ComEquity	Valor en libros / Patrimonio neto atribuible a accionistas.
VCRT_EF_FUND_EV	Valor de la empresa (Anual).
VCRT_EF_FUND_EV_B	Valor de la empresa (Semestral).
VCRT_EF_FUND_EV_Q	Valor de la empresa (Trimestral).
VCRT_EF_FUND_EV_RT	Valor de la empresa en tiempo real.
VCRT_EF_FUND_ShareholdersEquity	Patrimonio Neto (anual).
VCRT_EF_FUND_ShareCap	Capital.

VCRT_EF_FUND_OtherReserves	Otras reservas.
VCRT_EF_FUND_Dividends	Dividendos por acción anuales. Indica la parte de los beneficios que percibe cada acción. Es un dato muy utilizado por el inversor, ya que la política de dividendos, en ocasiones, significa una parte importante de los rendimientos del accionista. Este ratio bursátil es el cociente del dividendo pagado entre el número de acciones.
VCRT_EF_FUND_Dividends_B	Dividendos por acción semestrales.
VCRT_EF_FUND_Dividends_Q	Dividendos por acción trimestrales.
VCRT_EF_FUND_Liabilities	Total Pasivo.
VCRT_EF_FUND_NonCurrLiab	Pasivos no corrientes.
VCRT_EF_FUND_CurrLiab	Pasivos corrientes.
VCRT_EF_FUND_NonCurrBonds	Obligaciones no corrientes.
VCRT_EF_FUND_NonCurrBankDebs	Deudas no corrientes. Son deudas a largo plazo (más de un año), tanto por créditos de los bancos como por obligaciones u otros títulos emitidos.
VCRT_EF_FUND_OtherNonCurrFinancialLiab	Otros pasivos financieros no corrientes.
VCRT_EF_FUND_CurrBonds	Obligaciones corrientes.
VCRT_EF_FUND_CurrBankDebs	Deudas bancos corrientes.
VCRT_EF_FUND_Creditors	Acreedores.
VCRT_EF_FUND_OtherCurrFinancialLiab	Otros pasivos financieros corrientes.
VCRT_EF_FUND_OtherCurrLiab	Otros pasivos corrientes.
VCRT_EF_FUND_Assets	Total Activos.
VCRT_EF_FUND_NonCurrAssets	Activos no corrientes.
VCRT_EF_FUND_CurrAssets	Activos corrientes.
VCRT_EF_FUND_Inventory	Existencias.
VCRT_EF_FUND_Receivables	Realizable. Derechos de cobro, deudores comerciales etc...
VCRT_EF_FUND_Cash	Efectivo.
VCRT_EF_FUND_EPS	Beneficio por acción(BPA).
VCRT_EF_FUND_EPS_B	Beneficio por acción(BPA) semestral.
VCRT_EF_FUND_EPS_Q	Beneficio por acción(BPA) trimestral.
VCRT_EF_FUND_PER	Price Earning Ratio. Mide la relación entre el valor de mercado de una empresa y el beneficio total de la misma (anual).
VCRT_EF_FUND_PER_B	PER semestral.
VCRT_EF_FUND_PER_Q	PER trimestral.
VCRT_EF_FUND_PER_RT	PER en tiempo real. Es calculado cada 5 minuto.
VCRT_EF_FUND_PtoS	Precio por ventas (anual).
VCRT_EF_FUND_PtoS_B	Precio por ventas (semetral).
VCRT_EF_FUND_PtoS_Q	Precio por ventas (trimestral).
VCRT_EF_FUND_PtoS_RT	Precio por ventas (tiempo real). Se calcula cada 5 minutos.
VCRT_EF_FUND_PtoB	Precio por valor en libros (anual).
VCRT_EF_FUND_PtoB_RT	Precio por valor en libros (tiempo real). Se calcula cada 5 minutos.
VCRT_EF_FUND_PtoCF	Precio por flujo de caja (anual).
VCRT_EF_FUND_PtoCF_B	Precio por flujo de caja (semestral).
VCRT_EF_FUND_PtoCF_Q	Precio por flujo de caja (trimestral).
VCRT_EF_FUND_PtoCF_RT	Precio por flujo de caja (tiempo real). Se calcula cada 5

	minutos.
VCRT_EF_FUND_DivYield	Rendimiento/rentabilidad de dividendo (anual).
VCRT_EF_FUND_DivYield_B	Rendimiento/rentabilidad de dividendo (semetral).
VCRT_EF_FUND_DivYield_Q	Rendimiento/rentabilidad de dividendo (trimestral).
VCRT_EF_FUND_DivYield_RT	Rendimiento/rentabilidad de dividendo (tiempo real). Se calcula cada 5 mintuos.
VCRT_EF_FUND_PayOutR	Payout ratio (anual).
VCRT_EF_FUND_PayOutR_B	Payout ratio (semestral).
VCRT_EF_FUND_PayOutR_Q	Payout ratio (trimestral).
VCRT_EF_FUND_GrossMargin	Beneficios brutos sobre ventas (anual).
VCRT_EF_FUND_GrossMargin_B	Beneficios brutos sobre ventas (semestral).
VCRT_EF_FUND_GrossMargin_Q	Beneficios brutos sobre ventas (trimestral).
VCRT_EF_FUND_OpMargin	Beneficio operativo sobre ventas (anual).
VCRT_EF_FUND_OpMargin_B	Beneficio operativo sobre ventas (semestral).
VCRT_EF_FUND_OpMargin_Q	Beneficio operativo sobre ventas (trimestral).
VCRT_EF_FUND_PreTaxMargin	BAI sobre ventas (anual).
VCRT_EF_FUND_PreTaxMargin_B	BAI sobre ventas (semestral).
VCRT_EF_FUND_PreTaxMargin_Q	BAI sobre ventas (trimestral).
VCRT_EF_FUND_EfTaxRate	Tasa impositiva efectiva (anual).
VCRT_EF_FUND_EfTaxRate_B	Tasa impositiva efectiva (semestral).
VCRT_EF_FUND_EfTaxRate_Q	Tasa impositiva efectiva (trimestral).
VCRT_EF_FUND_ROA	Beneficio neto sobre activo (anual).
VCRT_EF_FUND_ROA_B	Beneficio neto sobre activo (semestral).
VCRT_EF_FUND_ROA_Q	Beneficio neto sobre activo (trimestral).
VCRT_EF_FUND_ROE	Beneficio netos sobre patrimonio neto (anual).
VCRT_EF_FUND_ROE_B	Beneficio netos sobre patrimonio neto (semestral).
VCRT_EF_FUND_ROE_Q	Beneficio netos sobre patrimonio neto (trimestral).
VCRT_EF_FUND_ROI	Beneficio sobre la inversión (anual).
VCRT_EF_FUND_ROI_B	Beneficio sobre la inversión (semestral).
VCRT_EF_CALF_DayCash	Efectivo negociado. Es la suma en el día del volumen negociado por el precio al que se ha realizado cada transacción.
VCRT_EF_CALF_MonthRotation	Rotación en meses. Número de meses que se ha tardado en operar un volumen igual al total de acciones.
VCRT_EF_CALF_DayHi	Máximo del día.
VCRT_EF_CALF_PToDayHI	Distancia porcentual al máximo del día.
VCRT_EF_CALF_WeekHi	Máximo de la semana.
VCRT_EF_CALF_PToWeekHi	Distancia porcentual al máximo de la semana.
VCRT_EF_CALF_MonthHi	Máximo del mes.
VCRT_EF_CALF_PToMonthHi	Distancia porcentual al máximo del mes.
VCRT_EF_CALF_52WeekHi	Máximo de las últimas 52 semanas.
VCRT_EF_CALF_PTo52WeekHi	Distancia porcentual al máximo de las últimas 52 semanas.
VCRT_EF_CALF_HistoricHi	Máximo del histórico.
VCRT_EF_CALF_PToHistoricHi	Distancia porcentual al máximo del histórico.
VCRT_EF_CALF_5YearsHi	Máximo de los últimos 5 años.
VCRT_EF_CALF_PTo5YearsHi	Distancia porcentual al máximo de los últimos 5 años.
VCRT_EF_CALF_CurrentYearHi	Máxima del año actual.
VCRT_EF_CALF_PToCYearHi	Distancia porcentual a la máxima del año actual.

VCRT_EF_CALF_DayLo	Mínimo del día.
VCRT_EF_CALF_PToDayLo	Distancia porcentual al mínimo del día.
VCRT_EF_CALF_WeekLo	Mínimo de la semana.
VCRT_EF_CALF_PToWeekLo	Distancia porcentual al mínimo de la semana.
VCRT_EF_CALF_MonthLo	Mínimo del mes.
VCRT_EF_CALF_PToMonthLo	Distancia porcentual al mínimo del mes.
VCRT_EF_CALF_52WeekLo	Mínimo de las últimas 52 semanas.
VCRT_EF_CALF_PTo52WeekLo	Distancia porcentual al mínimo de las últimas 52 semanas.
VCRT_EF_CALF_HistoricalLow	Mínimo del histórico.
VCRT_EF_CALF_PtoHistoricLow	Distancia porcentual al mínimo del histórico.
VCRT_EF_CALF_5YearsLo	Mínimo de los últimos 5 años.
VCRT_EF_CALF_PTo5YearsLo	Distancia porcentual al mínimo de los últimos 5 años.
VCRT_EF_CALF_CurrentYearLo	Mínimo del año actual.
VCRT_EF_CALF_PToCYearLo	Distancia porcentual al mínimo del año.
VCRT_EF_CALF_DayYield	Rentabilidad del día.
VCRT_EF_CALF_WeekYield	Rentabilidad de las semana.
VCRT_EF_CALF_MonthYield	Rentabilidad del mes.
VCRT_EF_CALF_52WeekYield	Rentabilidad de las últimas 52 semanas.
VCRT_EF_CALF_CurrentYYield	Rentabilidad del año actual.
VCRT_EF_CALF_5YearsYield	Rentabilidad de los últimos 5 años.
VCRT_EF_CALF_TradeDiferencial	Diferencial de negociación. Diferencia entre el volumen ejecutado en al compra y en al venta durante el día.
VCRT_EF_CALF_Beta	Coeficiente beta. Para el cálculo de este valor se toman los rendimientos semanales de los últimos 5 años del valor, y del índice de referencia (RefIndex Rv y Ri respectivamente. Se calcula la recta de regresión de dichos valores, usando Ri como variable de ordenadas y Rv como abscisas. La pendiente de dicha recta es Beta. En el caso de que no se disponga de 5 años de histórico se utiliza el periodo disponible siempre y cuando sea superior a 1 año. El cálculo y el refresco son semanales.
VCRT_EF_CALF_Delta	Delta de una opción.
VCRT_EF_CALF_Gamma	Gamma de una opción. Mide la velocidad de cambio de Delta
VCRT_EF_CALF_Theta	Mide la variación de una opción debido al paso del tiempo.
VCRT_EF_CALF_Vega	Mide la sensibilidad a la volatilidad del subyacente del precio de una opción.
VCRT_EF_CALF_Rho	Sensibilidad de una opción a los tipos de cambio.
VCRT_EF_CALF_Alfa	Coeficiente Alfa. Para el cálculo de este valor se toman los rendimientos semanales de los últimos 5 años del valor y del índice de referencia (RefIndex) Rv y Ri respectivamente. Se calcula la recta de regresión de dichos valores, usando Ri como variable de ordenadas y Rv como abscisas. La intersección de dicha recta con el eje de abscisas es el parámetro Alfa. En el caso de que no se disponga de 5 años de histórico se utiliza el periodo disponible siempre y cuando sea superior a 1 año. El cálculo y el refresco son semanales.
VCRT_EF_CALF_VaR	Valor en riesgo. Especifica el valor porcentual máximo

	que puede perder en una sesión con un 95% de certidumbre.
VCRT_EF_CALF_VaR99	Especifica el valor porcentual máximo que puede perder en una sesión con un 99% de certidumbre.
VCRT_EF_CALF_Liquidity	Liquidez. Un parámetro interesante para la evaluación de carteras es la liquidez de los valores que la integran. Entendemos por liquidez la facilidad que tenemos para cerrar una posición, o el inverso del coste que supone cerrar una posición debido a la falta de contrapartida suficiente.
VCRT_EF_CALF_AvPriceDay	Precio promedio del día.
VCRT_EF_CALF_AvPriceWeek	Precio promedio de la semana.
VCRT_EF_CALF_AvPriceMonth	Precio promedio del mes.
VCRT_EF_CALF_AvPriceQuarter	Precio promedio del trimestre.
VCRT_EF_CALF_AvPriceYear	Precio promedio del año.
VCRT_EF_CALF_AvMarketCapDay	Capitalización Mercado promedio del día.
VCRT_EF_CALF_AvMarketCapWeek	Capitalización Mercado promedio de la semana.
VCRT_EF_CALF_AvMarketCapMonth	Capitalización Mercado promedio del mes.
VCRT_EF_CALF_AvMarketCapQuarter	Capitalización Mercado promedio del trimestre.
VCRT_EF_CALF_AvMarketCapYear	Capitalización Mercado promedio del año.
VCRT_EF_RT_CapType	Clasificación por capitalización.
VCRT_EF_RT_YieldWeekly	Rentabilidad semanal.
VCRT_EF_RT_YieldMonthly	Rentabilidad mensual.
VCRT_EF_RT_YieldQuarterly	Rentabilidad trimestral.
VCRT_EF_RT_YieldBiannually	Rentabilida semestral.
VCRT_EF_RT_YieldYearly	Rentabilidad anual.
VCRT_EF_TECA_Support1	Primer soporte.
VCRT_EF_TECA_SupportViolated	Soporte violado.
VCRT_EF_TECA_PToSupport1	Distancia porcentual al primer soporte.
VCRT_EF_TECA_Support2	Segundo soporte.
VCRT_EF_TECA_PToSupport2	Distancia porcentual al segundo soporte.
VCRT_EF_TECA_Support3	Tercer soporte.
VCRT_EF_TECA_PToSupport3	Distancia porcentual al tercer soporte.
VCRT_EF_TECA_Resistance1	Primera resistencia.
VCRT_EF_TECA_ResistanceViolated	Resistencia violada.
VCRT_EF_TECA_PToResistance1	Distancia porcentual a la primera resistencia.
VCRT_EF_TECA_Resistance2	Segunda resistencia.
VCRT_EF_TECA_PToResistance2	Distancia porcentual a la segunda resistencia.
VCRT_EF_TECA_Resistance3	Tercera resistencia.
VCRT_EF_TECA_PToResistance3	Distancia porcentual a la tercera resistencia.
VCRT_EF_TECA_BullTLine	Línea de tendencia alcista.
VCRT_EF_TECA_CrossBullTLine	Linea de tendencia alcista cruzada.
VCRT_EF_TECA_BullTLine2	Segunda línea de tendencia alcista.
VCRT_EF_TECA_BullTLineSlope	Pendiente de la línea de tendencia alcista.
VCRT_EF_TECA_PToBullTLine	Distancia porcentual a la línea de tendencia alcista.
VCRT_EF_TECA_BearTLine	Línea de tendencia bajista.
VCRT_EF_TECA_CrossBearTLine	Línea de tendecia bajista cruzada.
VCRT_EF_TECA_BearTLine2	Segunda línea de tendencia.
VCRT_EF_TECA_BearTLineSlope	Pendiente de la línea de tendencia bajista.

VCRT_EF_TECA_PToBearTLine	Distancia porcentual a la línea de tendencia.
VCRT_EF_TECA_RegLine	Recta de regresión.
VCRT_EF_TECA_RegSlope	Pendiente de la recta de regresión.
VCRT_EF_TECA_TopRegChannel	Límite superior del canal de regresión.
VCRT_EF_TECA_BotRegChannel	Límite inferior del canla de regresión.
VCRT_EF_TECA_CrossTopRegChannel	Cruzado el límite superior del canal de regresión.
VCRT_EF_TECA_CrossBotRegChannel	Cruzado el límite inferior del canal de regresión.
VCRT_EF_TECA_RegLineParams	Parámetros para pintar la línea y el canal de regresión.
VCRT_EF_TECA_PToRegLine	Distancia porcentual a la recta de regresión.
VCRT_EF_TECA_PToTopRegCh	Distancia porcentual a la línea superior del canal de regresión.
VCRT_EF_TECA_PToBotRegCh	Distancia porcentual a la línea inferior del canla de regresión.
VCRT_EF_TECA_PercentGap	Gap porcentual. Si se produce un hueco entre la apertura del día y el cierre del día anterior, este campo muestra la distancia porcentual.
VCRT_EF_TECA_DoubleTop	Figura chartista de doble techo.
VCRT_EF_TECA_DoubleBottom	Figura chartista de doble suelo.
VCRT_EF_TECA_HeadShoulders	Figura chartista hombro-cabeza-hombro.
VCRT_EF_TECA_BreakingFloor	Rotura de soporte.
VCRT_EF_TECA_FibonacciR1	Primer retroceso de Fibonacci.
VCRT_EF_TECA_CrossFibR1	Primer retroceso cruzado.
VCRT_EF_TECA_PToFibonacciR1	Distancia porcentual al primer retroceso de Fibonacci.
VCRT_EF_TECA_FibonacciR2	Segundo retroceso de Fibonacci.
VCRT_EF_TECA_CrossFibR2	Segundo retroceso cruzado.
VCRT_EF_TECA_PToFibonacciR2	Distancia porcentual al segundo retroceso de Fibonacci.
VCRT_EF_TECA_FibonacciR3	Tercer retroceso de Fibonacci.
VCRT_EF_TECA_CrossFibR3	Tercer retroceso cruzado.
VCRT_EF_TECA_PToFibonacciR3	Distancia porcentual al tercer retroceso de Fibonacci.
VCRT_EF_TECA_FibonacciR4	Cuarto retroceso de fibonacci.
VCRT_EF_TECA_CrossFibR4	Cuarto retroceso cruzado.
VCRT_EF_TECA_PToFibonacciR4	Distancia porcentual al cuarto retroceso de Fibonacci.
VCRT_EF_TECA_GannR1	Primer retroceso de Gann.
VCRT_EF_TECA_PToGannR1	Distancia porcentual al pimer retroceso de Gann.
VCRT_EF_TECA_GannR2	Segundo retroceso de Gann.
VCRT_EF_TECA_PToGannR2	Distancia porcentual al segundo retroceso de Gann.
VCRT_EF_TECA_GannR3	Tercer retroces de Gann.
VCRT_EF_TECA_PToGannR3	Distancia porcentual al tercer retroceso de Gann.
VCRT_EF_TECA_GannR4	Cuarto retroceso de Gann.
VCRT_EF_TECA_PToGannR4	Distancia porcentual al cuarto retroceso de Gann.
VCRT_EF_TECA_ElliotWN	Número de onda de Elliot.
VCRT_EF_TECA_WMagnitude	Amplitud de onda.
VCRT_EF_TECA_AvWMagnitude	Amplitud media de onda.
VCRT_EF_TECA_CTrendMagnitude	Amplitud de la tendencia actual.
VCRT_EF_TECA_AvTrendMagnitude	Amplitud media de tendencia.
VCRT_EF_TECA_Hammer	Vela del tipo sombrilla en la que se espera que una tendencia bajista termine.
VCRT_EF_TECA_HangingMan	Vela del tipo sombrilla en la que se espera que una tendencia alcista termine.

VCRT_EF_TECA_InvertedHammer	Vela con cuerpo real pequeño y sombra superior grande que pretende marcar el final de una tendencia bajista.
VCRT_EF_TECA_BullishHarami	Patrón alcista compuesto por 2 velas. La primera es una vela grande. La segunda es una vela del tipo peonza que está dentro del cuerpo real de la vela anterior.
VCRT_EF_TECA_BearishHarami	Patrón bajista compuesto por 2 velas. La primera es una vela grande. La segunda es una vela del tipo peonza que está dentro del cuerpo real de la vela anterior.
VCRT_EF_TECA_ShootingStar	Vela del tipo estrella en la que se espera que una tendencia alcista termine.
VCRT_EF_TECA_EveningStar	Patrón del tipo estrella formado por 3 velas. La primera es una vela blanca grande. La segunda es una estrella. La última confirma el cambio de tendencia. Con este patrón se espera que una tendencia alcista termine.
VCRT_EF_TECA_MorningStar	Patrón del tipo estrella formado por 3 velas. La primera es una vela negra grande. La segunda es una estrella. La última confirma el cambio de tendencia. Con este patrón se espera que una tendencia bajista termine.
VCRT_EF_TECA_DarkCloudCover	Patrón de cambio bajista compuesto por 2 velas. La primera es una vela blanca grande. La apertura de la segunda vela aparece por encima del máximo de la vela anterior y su cierre desciende más allá de la mitad del cuerpo real de la vela blanca anterior.
VCRT_EF_TECA_PiercingLine	Patrón de cambio alcista compuesto por 2 velas. La primera es una vela negra grande. La apertura de la segunda vela aparece por debajo del mínimo de la vela anterior y su cierre asciende más allá de la mitad del cuerpo real de la vela negra anterior.
VCRT_EF_TECA_BullishEngulfing	Patrón de cambio alcista compuesto por 2 velas. La primera es una vela negra. La segunda vela es una vela blanca grande que debe envolver a la vela anterior.
VCRT_EF_TECA_BearishEngulfing	Patrón de cambio bajista compuesto por 2 velas. La primera es una vela blanca. La segunda es una vela negra grande que debe envolver a la vela anterior.
VCRT_EF_TECA_ThreeWhiteSoldiers	Patrón de cambio alcista compuesto por 3 velas. Cada vela abre dentro o cerca de la vela blanca anterior. Además, cada vela debería cerrar en máximos.
VCRT_EF_TECA_ThreeBlackCrows	Patrón de cambio bajista compuesto por 3 velas.
VCRT_EF_IND_RSI	RSI de 14 sesiones.
VCRT_EF_IND_TendRSI	Tendencia del RSI.
VCRT_EF_IND_MACD	MACD 12 sesiones. El MACD es la diferencia de dos medias exponenciales: Media1 con periodo 12 Media2 con periodo 26 MACD=AVExp(12) - AVExp(26)
VCRT_EF_IND_MACDSig	Línea de señal de actuación MACD.
VCRT_EF_IND_TendMACD	Tendencia del MACD.
VCRT_EF_IND_StochasticSK	Estocástico (línea SK).
VCRT_EF_IND_StochasticSD	Estocástico (línea SD).
VCRT_EF_IND_TendStochastic	Tendencia del Estocástico.
VCRT_EF_IND_BollingerCL10	Bollinger línea central 10 de sesiones.
VCRT_EF_IND_BollingerUpL10	Bollinger línea superior 10 de sesiones.
VCRT_EF_IND_BollingerDownL10	Bollinger línea inferior 10 de sesiones.
VCRT_EF_IND_BollingerCL20	Bollinger línea central 20 de sesiones.
VCRT_EF_IND_BollingerUpL20	Bollinger línea superior 10 de sesiones.

VCRT_EF_IND_BollingerDownL20	Bollinger línea inferior 10 de sesiones.
VCRT_EF_IND_AV15	Media móvil de 15 sesiones.
VCRT_EF_IND_TendAV15	Tendencia de la media móvil de 15 sesiones.
VCRT_EF_IND_PToAV15	Distancia porcentual a la media de 15 sesiones.
VCRT_EF_IND_AV50	Media móvil de 50 sesiones.
VCRT_EF_IND_TendAV50	Tendecia de la media móvil de 50 sesiones.
VCRT_EF_IND_PToAV50	Distancia porcentual a la media de 50 sesiones.
VCRT_EF_IND_AV100	Media móvil de 100 sesiones.
VCRT_EF_IND_TendAV100	Tendencia a la media móvil de 100 sesiones.
VCRT_EF_IND_PToAV100	Distancia porcentual a la media de 100 sesiones.
VCRT_EF_IND_AV200	Media móvil de 200 sesiones.
VCRT_EF_IND_TendAV200	Tendencia de la media móvil de 200 sesiones.
VCRT_EF_IND_PToAV200	Distancia porcentual a la media móvil de 200 sesiones.
VCRT_EF_IND_DifAv15Av50	Diferencia porcentual entre la media de 15 y de 50 sesiones.
VCRT_EF_IND_DifAv50Av200	Diferencia porcentual entre la media de 50 y 200 sesiones.
VCRT_EF_IND_DayVol	Volumen del día.
VCRT_EF_IND_WeekVol	Volumen de la semana.
VCRT_EF_IND_MonthVol	Volumen del mes.
VCRT_EF_IND_52WeekVol	Volumen de las últimas 52 semanas.
VCRT_EF_IND_Av52WVol	Volumen medio de las últimas 52 semanas.
VCRT_EF_IND_TendAv52WVol	Tendencia del volumen medio de las últimas 52 semanas.
VCRT_EF_IND_CurrentYVol	Volumen año actual.
VCRT_EF_IND_AvVol15	Media del volumen de 15 sesiones.
VCRT_EF_IND_TendAvVol15	Tendencia Media móvil de volumen de 15 sesiones.
VCRT_EF_IND_RelativeVol	Diferencia entre el volumen de la sesión actual con media móvil de volumen de 15 sesiones.
VCRT_EF_IND_ComparativeVol	Diferencial de volumen (calcula la diferencia entre una comparativa del volumen de un mes con respecto a las últimas 52 semanas con idéntica comparativa calculada para el mismo mes del año anterior).
VCRT_EF_IND_DayVolatility	Volatilidad del día (La volatilidad es una indicación del riesgo debido a las fluctuaciones del valor). Este campo se calcula de forma similar al indicador de Indice de Volatilidad pero ponderada por el precio del valor: Definimos: Ci-1 = Cierre de la anterior barra MAXi = Máximo de la barra MINi = Mínimo de la barra Calculamos: a = | MAXi - MINi| b = | Ci-1 - MAXi| c = | Ci-1 - MINi| TRi = 100 * max (a, b, c)/ Ci-1 Lo ponderamos por el precio de cierre de la anterior barra. El primer valor de volatilidad es la media de los TRi

	para el periodo 'p' indicado. Las siguientes se calculan: Vi = (TRi + Vi-1 * (p - 1)) /p Para los diferentes valores de volatilidad se establecen los siguientes periodos y barras a utilizar: Dia => 100 barras de 15 minutos Semana => 500 barras de 15 minutos Mes => 23 barras de 1 día 52 Semanas => 250 barras de 1 día 5 años => 1250 barras de 1 día
VCRT_EF_IND_DayVolatilityAlert	Alerta de volatilidad diaria. Si TRi > 3 * Vi-1 en la volatilidad del día .
VCRT_EF_IND_WeekVolatility	Volatilidad de la semana.
VCRT_EF_IND_MonthVolatility	Volatilidad del mes.
VCRT_EF_IND_52WeekVolatility	Volatilidad de las últimas 52 semanas.
VCRT_EF_IND_5YearsVolatility	Volatilidad de los últimos 5 años.
VCRT_EF_IND_ADX	Movimiento Direccional 14 sesiones.
VCRT_EF_IND_TendADX	Tendencia del ADX de 14 sesiones.
VCRT_EF_IND_PositiveDi	DI + de 14 sesiones.
VCRT_EF_IND_TendPositiveDi	Tendencia del DI+ de 14 sesiones.
VCRT_EF_IND_NegativeDi	DI- de 14 sesiones.
VCRT_EF_IND_TendNegativeDi	Tendencia del DI- de 14 sesiones.
VCRT_EF_IND_AccumDis	AccumDis
VCRT_EF_IND_StdDesviation	Desviación estándar. Se calcula la desviación estándar de las rentabilidades semanales de los 5 últimos años. En caso de no disponer de 5 años, se utiliza el tiempo de que se disponga, pero como mínimo de 1 año. La frecuencia de cálculo y refresco es semanal.
VCRT_EF_PORT_PHYield	
VCRT_EF_PORT_PDYield	Rentabilidad porcentual del día actual.
VCRT_EF_PORT_PWYield	Rentabilidad porcentual de la semana.
VCRT_EF_PORT_PMYield	Rentabilidad del mes.
VCRT_EF_PORT_P52WYield	Rentabilidad de las últimas 52 semanas.
VCRT_EF_PORT_PCYYield	Rentabilidad porcentual del año actual.
VCRT_EF_PORT_P3YYield	Rentabilidad porcentual de los últimos 3 años.
VCRT_EF_PORT_P5YYield	Rentabilidad porcentual de los últimos 5 años.
VCRT_EF_PORT_PPIndex	Valor histórico de la posición actual de una cartera.
VCRT_EF_PORT_PVaR	VaR (Value at Risk) de la cartera.
VCRT_EF_PORT_PVolume	Volumen (número de títulos).
VCRT_EF_PORT_PPrice	Precio (por título).
VCRT_EF_PORT_PValue	Valor de mercado de las posiciones abiertas de una cartera.
VCRT_EF_PORT_PCash	Efectivo disponible de la cartera.
VCRT_EF_PORT_PCredit	Ingresos de una transacción.
VCRT_EF_PORT_PBalance	Es el Balance (ingresos menos gastos). Indica las ganancias o pérdidas de las operaciones.
VCRT_EF_PORT_PNumTransacts	Número de transacciones realizadas.
VCRT_EF_PORT_PNumPositions	Número de posiciones de la cartera.
VCRT_EF_PORT_AvgBuyPrice	Precio medio de compra de una posición. Si se ha

	comprado una misma acción en dos operaciones a diferentes precios distintos, este campo muestra el precio medio en función del número de títulos comprado en cada operación.
VCRT_EF_PORT_PShort	Indica si la posición está vendida o no. Actualmente no se permiten posiciones a la baja en la cartera pero se inclulye en previsión de futuras versiones.
VCRT_EF_PORT_PDividends	Dividendos percibidos (acumulados hasta la fecha).
VCRT_EF_PORT_PNumDividends	Dividendos percibidos (acumulados hasta la fecha).
VCRT_EF_PORT_PNumEarnings	Número de operaciones cerradas.
VCRT_EF_PORT_PAccount	Cuenta a la que va referida una transacción.
VCRT_EF_PORT_PTransactType	Tipo de transacción. Puede ser compra, venta, ingreso, reembolso, comisión o dividendo.
VCRT_EF_PORT_PDebit	Gastos de una transacción.
VCRT_EF_PORT_PFees	Comisiones.
VCRT_EF_PORT_PGain	Garantías realizadas.
VCRT_EF_PORT_PGLatentes	Garantías latentes.
VCRT_EF_PORT_Patrimonio	Patrimonio de una cartera (PValue + PCash)
VCRT_EF_PORT_PReturn	Ganancia porcentual histórica.
VCRT_EF_INDA_DayGainers	Número de empresas que suben día (empresas que suben de todos los índices y mercados) .
VCRT_EF_INDA_DayLosers	Número de empresas que bajan día (empresas que suben de todos los índices y mercados).
VCRT_EF_INDA_DayRepeat	Número de empresas que repiten en el día(empresas que suben de todos los índices y mercados) .

ENUMERACIONES VCDataSource

A continuación se indican los valores que pueden tomar un tipo de variable de la librería **VCDataSource**.

EnumVCDSOrderType. Indica el tipo de orden que se ha ejecutado en el sistema (ver VCDS_SystemOrder y VCDS_Trade).

Valor	Descripción
VCDS_OT_Market	Orden a mercado.
VCDS_OT_Limit	Orden limitada.
VCDS_OT_Stop	Orden de stop.

EnumVCDSOrderSide. Indica si la orden que ha generado el negocio ha sido una compra o una venta (ver VCDS_Trade).

Valor	Descripción
VCDS_OS_Unknown	Desconocido (no especificado).
VCDS_OS_Buy	Compra.
VCDS_OS_Sell	Venta.

EnumVCDSDataSourceType. Indica el tipo de fuente (ver VCDS_DataSource).

Valor	Descripción
VCDS_DT_Any	Sin especificar.
VCDS_DT_DataSerie	La fuente es una serie (VCDS_DataSerie).
VCDS_DT_Indicator	La fuente es un indicador (VCDS_Indicator).
VCDS_DT_System	La fuente es un sistema (VCDS_System).

EnumVCDSCompressionType. Indica el tipo de compresión de una serie. Para las fuentes de ticks, las unidades de la compresión indican el número de ticks a comprimir en cada barra. Para el resto de tipos de compresión, las unidades indican el número de minutos/días/semanas/meses.

Valor	Descripción
VCDS_CT_Ticks	Compresión de ticks.
VCDS_CT_Minutes	Compresión de minutos.
VCDS_CT_Days	Compresión de días.
VCDS_CT_Weeks	Compresión de semanas.
VCDS_CT_Months	Compresión de meses.

EnumVCDSContextWorkMode. Indica el modo de contexto activo. Cuando se crean series de datos, estas pueden compartir o no la escala temporal. Compartir la escala de tiempo es útil si se desea comparar dos series de datos, insertar un indicador que tenga como padres dos series de datos, etc.

Las series que comparten contexto, comparten el contexto por defecto, que es el primer contexto creado que no sea de una serie con compresión de ticks o, si solo existen series de ticks, el contexto de la primera serie de ticks.

Valor	Descripción
VCDS_CWM_Multiple	Las series que se crean lo hacen en contextos independientes.
VCDS_CWM_Shared	Las series comparten un mismo contexto, el contexto por defecto.

EnumVCDSInstrumentType. Tipo de instrumento (ver VCDS_SymbolInfo).

Valor	Descripción
VCDS_IT_Currencies	Divisas.
VCDS_IT_Future	Futuros.
VCDS_IT_Index	Índices.
VCDS_IT_Mixed	Misceláneo.
VCDS_IT_Option	Opciones.
VCDS_IT_Fund	Fondos.
VCDS_IT_Warrant	Warrants.
VCDS_IT_Stock	Valores.

EnumVCDSSystemEvent. Informa del suceso acaecido en el evento OnSystemEvent.

Valor	Descripción
VCDS_SE_NewTrade	Se ha generado un nuevo negocio.
VCDS_SE_NewOrder	Se ha ejecutado una nueva orden.
VCDS_SE_ChangedPosition	Se ha producido un cambio de posición.

EnumVCDSStatisticVariable. Indica la variable estadística.

Valor	Variable estadística en VBA (SistemaApp)	Descripción
VC_SV_Gross	svGross	Ganancia Bruta.
VC_SV_GrossProfit	svGrossProfit	Ganancia Bruta (Acumulada).
VC_SV_Stdev_Gross	svStdev_Gross	Ganancia Bruta (Desviación Típica).

VC_SV_Avg_Gross	svAvg_Gross	Ganancia Bruta (Media).
VC_SV_CV_Gross	svCV_Gross	Ganancia Bruta (Coeficiente de Variación).
VC_SV_StdevPAvg_Gross	svStdevPAvg_Gross	Ganancia Bruta (+1 Desv).
VC_SV_StdevMAvg_Gross	svStdevMAvg_Gross	Ganancia Bruta (-1 Desv).
VC_SV_Max_Gross	svMax_Gross	Ganancia Bruta (Máximo).
VC_SV_Min_Gross	svMin_Gross	Ganancia Bruta (Mínimo).
VC_SV_Select_Gross	svSelect_Gross	Ganancia Bruta (Select).
VC_SV_Outlier_Gross	svOutlier_Gross	Ganancia Bruta (Outliers).
VC_SV_Net	svNet	Ganancia Neta.
VC_SV_NetProfit	svNetProfit	Ganancia Neta (Acumulada).
VC_SV_Stdev_Net	svStdev_Net	Ganancia Neta (Desviación Típica).
VC_SV_Avg_Net	svAvg_Net	Ganancia Neta (Media).
VC_SV_CV_Net	svCV_Net	Ganancia Neta (Coeficiente de Variación).
VC_SV_StdevPAvg_Net	svStdevPAvg_Net	Ganancia Neta (+1 Desv).
VC_SV_StdevMAvg_Net	svStdevMAvg_Net	Ganancia Neta (-1 Desv).
VC_SV_Max_Net	svMax_Net	Ganancnia Neta (Máximo).
VC_SV_Min_Net	svMin_Net	Ganancia Neta (Mínimo).
VC_SV_Select_Net	svSelect_Net	Ganancia Neta (Select).
VC_SV_Outlier_Net	svOutlier_Net	Ganancia Neta (Outliers).
VC_SV_Stdev_NetProfit	svStdev_NetProfit	Ganancia Neta Acumulada (Desviación Típica).
VC_SV_Avg_NetProfit	svAvg_NetProfit	Ganancia Neta Acumulada (Media).
VC_SV_CV_NetProfit	svCV_NetProfit	Ganancia Neta Acumulada (Coeficiente de Variación).
VC_SV_StdevPAvg_NetProfit	svStdevPAvg_NetProfit	Ganancia Neta Acumulada (+1 Desv).
VC_SV_StdevMAvg_NetProfit	svStdevMAvg_NetProfit	Ganancia Neta Acumulada (-1 Desv).
VC_SV_Max_NetProfit	svMax_NetProfit	Ganancia Neta Acumulda (Máximo).
VC_SV_Min_NetProfit	svMin_NetProfit	Ganancia Neta Acumulada (Mínimo).
VC_SV_Select_NetProfit	svSelect_NetProfit	Selecto Neto (Acumulado).
VC_SV_LongShortRatio	svLongShortRatio	Ratio: Largo/Corto.
VC_SV_WinnersLosersRatio	svWinnersLosersRatio	Ratio: Positivo/Negativo.
VC_SV_ProfitFactor	svProfitFactor	Factor de ganancia.
VC_SV_Adjusted_ProfitFactor	svAdjusted_ProfitFactor	Factor de ganancia ajustado (PRR).
VC_SV_Select_ProfitFactor	svSelect_ProfitFactor	Factor de ganancia (Select).

VC_SV_Fiability	svFiability	Fiabilidad.
VC_SV_TradesCount	svTradesCount	Número de negocios.
VC_SV_Open_TradesCount	svOpen_TradesCount	Número de negocios. (Pos. Abiertas).
VC_SV_LinearRegression	svLinearRegression	Recta de regresión.
VC_SV_Adjusted_GrossProfit	svAdjusted_GrossProfit	Ganancia Bruta Ajustada.
VC_SV_Adjusted_NetOverall	svAdjusted_NetOverall	Ganancia Neta Ajustada.
VC_SV_Drawdown	svDrawdown	DrawDown.
VC_SV_NetDrawdown	svNetDrawdown	DrawDown (Acumulado).
VC_SV_Avg_Drawdown	svAvg_Drawdown	DrawDown (Media).
VC_SV_Stdev_Drawdown	svStdev_Drawdown	DrawDown (Desviación Típica).
VC_SV_StdevPAvg_Drawdown	svStdevPAvg_Drawdown	DrawDown (-1 Desv).
VC_SV_StdevMAvg_Drawdown	svStdevMAvg_Drawdown	DrawDown (-1 Desv).
VC_SV_CV_Drawdown	svCV_Drawdown	DrawDown (Coeficiente de Variación).
VC_SV_Max_Drawdown	svMax_Drawdown	DrawDown (Máximo).
VC_SV_Min_Drawdown	svMin_Drawdown	DrawDown (Mínimo).
VC_SV_Runup	svRunup	Runup.
VC_SV_NetRunup	svNetRunup	Runup (Acumulado).
VC_SV_Avg_Runup	svAvg_Runup	Runup (Media).
VC_SV_Stdev_Runup	svStdev_Runup	Runup (Desviación Típica).
VC_SV_StdevPAvg_Runup	svStdevPAvg_Runup	Runup (+1 Desv).
VC_SV_StdevMAvg_Runup	svStdevMAvg_Runup	Runup (-1 Desv).
VC_SV_CV_Runup	svCV_Runup	Runup (Coeficiente de Variación).
VC_SV_Max_Runup	svMax_Runup	Runup (Máximo).
VC_SV_Min_Runup	svMin_Runup	Runup (Mínimo).
VC_SV_Adjusted_NetOverall_DIV_Min_Profit	svAdjusted_NetOverall_DIV_Min_Profit	Ratio: Ganancia Neta Ajustada/Ganancia Mínima.
VC_SV_NetProfit_DIV_Min_Profit	svNetProfit_DIV_Min_Profit	Ratio: Ganancia Neta Acumulada/Ganancia Neta Mínima.
VC_SV_Adjusted_NetOverall_DIV_Min_Drawdown	svAdjusted_NetOverall_DIV_Min_Drawdown	Ganancia Neta Ajustada/Drawdown Mínimo.
VC_SV_NetProfit_DIV_Min_Drawdown	svNetProfit_DIV_Min_Drawdown	Ganancia Neta acumulada/Drawdown Mínimo.
VC_SV_Open_BarCount	svOpen_BarCount	Número de barras (Pos. abiertas).
VC_SV_BarCount	svBarCount	Número de barras.
VC_SV_Acc_BarCount	svAcc_BarCount	Número de barras (Acumulado).
VC_SV_Avg_BarCount	svAvg_BarCount	Número de barras (Media).

VC_SV_Max_BarCount	svMax_BarCount	Número de barras (Máximo).
VC_SV_Min_BarCount	svMin_BarCount	Número de barras (Mínimo).
VC_SV_Stdev_BarCount	svStdev_BarCount	Número de barras (Desviación Típica).
VC_SV_StdevPAvg_BarCount	svStdevPAvg_BarCount	Número de barras (+1 Desv).
VC_SV_StdevMAvg_BarCount	svStdevMAvg_BarCount	Número de barras (- 1 Desv).
VC_SV_BarsBetweenTrades	svBarsBetweenTrades	Barras entre negocios.
VC_SV_Acc_BarsBetweenTrades	svAcc_BarsBetweenTrades	Barras entre negocios (Acumulado).
VC_SV_Avg_BarsBetweenTrades	svAvg_BarsBetweenTrades	Barras entre negocios (Media).
VC_SV_Max_BarsBetweenTrades	svMax_BarsBetweenTrades	Barras entre negocios (Máximo).
VC_SV_Min_BarsBetweenTrades	svMin_BarsBetweenTrades	Barras entre negocios (Mínimo).
VC_SV_Stdev_BarsBetweenTrades	svStdev_BarsBetweenTrades	Barras entre negocios (Desviacón Típica).
VC_SV_StdevPAvg_BarsBetweenTrades	svStdevPAvg_BarsBetweenTrades	Barras entre negocios (+1 Desv).
VC_SV_StdevMAvg_BarsBetweenTrades	svStdevMAvg_BarsBetweenTrades	Barras entre negocios (-1 Desv).
VC_SV_EntryPrice	svEntryPrice	Punto de entrada.
VC_SV_ExitPrice	svExitPrice	Punto de salida.
VC_SV_EntryDate	svEntryDate	Fecha de entrada.
VC_SV_ExitDate	svExitDate	Fecha de salida.
VC_SV_EntryBar	svEntryBar	Barra de entrada.
VC_SV_ExitBar	svExitBar	Barra de salida.
VC_SV_LongShort	svLongShort	Compra/Venta (Pos. Abierta).
VC_SV_DaysSinceMaxProfit	svDaysSinceMaxProfit	Días desde ganancia máxima.
VC_SV_DaysSinceMinProfit	svDaysSinceMinProfit	Días desde ganancia mínima.
VC_SV_Acc_DaysSinceMaxProfit	svAcc_DaysSinceMaxProfit	Días desde ganancia máxima acumulada.
VC_SV_Acc_DaysSinceMinProfit	svAcc_DaysSinceMinProfit	Días desde ganancia mínima acumulada.
VC_SV_Avg_DaysSinceMaxProfit	svAvg_DaysSinceMaxProfit	Días desde ganancia máxima media.
VC_SV_Avg_DaysSinceMinProfit	svAvg_DaysSinceMinProfit	Días desde ganancia mínima media.
VC_SV_RetracementSinceMaxProfit	svRetracementSinceMaxProfit	Retroceso desde ganancia máxima.
VC_SV_ImprovementSinceMinProfit	svImprovementSinceMinProfit	Mejora desde ganancia mínima.
VC_SV_Efficiency	svEfficiency	Eficiencia.

VC_SV_EntryEfficiency	svEntryEfficiency	Eficiencia de entrada.
VC_SV_ExitEfficiency	svExitEfficiency	Eficiencia de salida.
VC_SV_NetEfficiency	svNetEfficiency	Eficiencia acumulada.
VC_SV_NetEntryEfficiency	svNetEntryEfficiency	Eficiencia de entrada acumulada.
VC_SV_NetExitEfficiency	svNetExitEfficiency	Eficiencia de salida acumulada.
VC_SV_Avg_Efficiency	svAvg_Efficiency	Eficiencia media.
VC_SV_Avg_EntryEfficiency	svAvg_EntryEfficiency	Eficiencia de entrada (Media).
VC_SV_Avg_ExitEfficiency	svAvg_ExitEfficiency	Eficiencia de salida (Media).
VC_SV_Stdev_Efficiency	svStdev_Efficiency	Eficiencia (Desviación Típica).
VC_SV_Stdev_EntryEfficiency	svStdev_EntryEfficiency	Eficiencia de entrada (Desviación Típica).
VC_SV_Stdev_ExitEfficiency	svStdev_ExitEfficiency	Efic. de sal. (Des. Típica)
VC_SV_Max_Efficiency	svMax_Efficiency	Eficiencia Máxima.
VC_SV_Max_EntryEfficiency	svMax_EntryEfficiency	Eficiencia de entrada (Máxima).
VC_SV_Max_ExitEfficiency	svMax_ExitEfficiency	Eficiencia de salida (Máxima).
VC_SV_Min_Efficiency	svMin_Efficiency	Eficiencia Mínima.
VC_SV_Min_EntryEfficiency	svMin_EntryEfficiency	Eficiencia de entrada (Mínima).
VC_SV_Min_ExitEfficiency	svMin_ExitEfficiency	Eficiencia de salida (Mínima).
VC_SV_CV_Efficiency	svCV_Efficiency	Eficiencia (Coeficiente de Variación).
VC_SV_CV_EntryEfficiency	svCV_EntryEfficiency	Eficiencia de entrada (C. Var)
VC_SV_CV_ExitEfficiency	svCV_ExitEfficiency	Eficiencia de salida (Coeficiente de variación).
VC_SV_StdevPAvg_Efficiency	svStdevPAvg_Efficiency	Eficiencia (+1 Desv).
VC_SV_StdevPAvg_EntryEfficiency	svStdevPAvg_EntryEfficiency	Eficiencia de entrada (+1 Desv).
VC_SV_StdevPAvg_ExitEfficiency	svStdevPAvg_ExitEfficiency	Eficiencia de salida (+1 Desv).
VC_SV_StdevMAvg_Efficiency	svStdevMAvg_Efficiency	Eficiencia (- Desv).
VC_SV_StdevMAvg_EntryEfficiency	svStdevMAvg_EntryEfficiency	Eficiencia de entrada (+1 Desv).
VC_SV_StdevMAvg_ExitEfficiency	svStdevMAvg_ExitEfficiency	Eficiencia de salida (-1 Desv).
VC_SV_Commisions	svCommisions	Comisiones.
VC_SV_NetCommisions	svNetCommisions	Comisiones (Acumulado).
VC_SV_NetCommisions_DIV_NetProfit	svNetCommisions_DIV_NetProfit	Ratio: Comisiones acumuladas/Ganancia neta acumulada.
VC_SV_MarketProfit	svMarketProfit	Ganancia del mercado.

VC_SV_NetMarketProfit	svNetMarketProfit	Ganancia del mercado (Acumulada).
VC_SV_SystemMarketRatio	svSystemMarketRatio	Ratio: Sistema/Mercado.
VC_SV_SharesCount	svSharesCount	Número de contratos/acciones.
VC_SV_Max_SharesCount	svMax_SharesCount	Número de contratos/acciones (Máximo).
VC_SV_PercentageInMarketTime	svPercentageInMarketTime	Porcentaje de tiempo en mercado.
VC_SV_InMarketTime	svInMarketTime	Tiempo negociado.
VC_SV_OverallTime	svOverallTime	Tiempo total.
VC_SV_TradeTime	svTradeTime	Tiempo de los negocios.
VC_SV_Max_TradeTime	svMax_TradeTime	Tiempo de los negocios (Máximo).
VC_SV_Acc_TradeTime	svAcc_TradeTime	Tiempo de los negocios (Mínimo).
VC_SV_Avg_TradeTime	svAvg_TradeTime	Tiempo de los negocios (Media).
VC_SV_TimeBetweenTrades	svTimeBetweenTrades	Tiempo entre los negocios.
VC_SV_Max_TimeBetweenTrades	svMax_TimeBetweenTrades	Tiempo entre los negocios (Máximo).
VC_SV_Acc_TimeBetweenTrades	svAcc_TimeBetweenTrades	Tiempo entre los negocios (Acumulado).
VC_SV_Avg_TimeBetweenTrades	svAvg_TimeBetweenTrades	Tiempo entre los negocios (Media).
VC_SV_TradesByYear_InMarketTime	svTradesByYear_InMarketTime	Negocios por año (Tiempo negociado).
VC_SV_TradesByMonth_InMarketTime	svTradesByMonth_InMarketTime	Negocios por mes (Tiempo negociado).
VC_SV_TradesByWeek_InMarketTime	svTradesByWeek_InMarketTime	Negocios por semana (Tiempo negociado).
VC_SV_TradesByDay_InMarketTime	svTradesByDay_InMarketTime	Negocios por día (Tiemp negociado).
VC_SV_TradesByYear_OverallTime	svTradesByYear_OverallTime	Negocios por año (Tiempo total).
VC_SV_TradesByMonth_OverallTime	svTradesByMonth_OverallTime	Negocios por mes (Tiempo total).
VC_SV_TradesByWeek_OverallTime	svTradesByWeek_OverallTime	Negocios por semana (Tiempo total).
VC_SV_TradesByDay_OverallTime	svTradesByDay_OverallTime	Negocios por día (Tiempo total).
VC_SV_RinaIndex	svRinaIndex	RINA Index.
VC_SV_Open_Profit	svOpen_Profit	Ganancnia Neta (Pos. abiertas).
VC_SV_Open_NetProfit	svOpen_NetProfit	Ganancia Neta Acumulada (Pos. abiertas).
VC_SV_Open_NetProfit_DIV_NetProfit	svOpen_NetProfit_DIV_NetProfit	anancia Neta Acumulada (Pos. abiertas)/Ganancia Neta Acumulada.

VC_SV_Avg_Open_Profit	svAvg_Open_Profit	Ganancia Neta Media (Pos. abiertas).
VC_SV_Avg_Open_Profit_DIV_Avg_Profit	svAvg_Open_Profit_DIV_Avg_Profit	Ratio: Ganancia Neta Media (Posiciones abiertas)/Ganancia Neta Media.
VC_SV_ProfitByYear_InMarketTime	svProfitByYear_InMarketTime	Ganancia por año (Tiempo negociado).
VC_SV_ProfitByMonth_InMarketTime	svProfitByMonth_InMarketTime	Ganancia por mes (Tiempo negociado).
VC_SV_ProfitByWeek_InMarketTime	svProfitByWeek_InMarketTime	Ganancia por semana (Tiempo negociado).
VC_SV_ProfitByDay_InMarketTime	svProfitByDay_InMarketTime	Ganancia por día (Tiempo negociado).
VC_SV_ProfitByYear_OverallTime	svProfitByYear_OverallTime	Gan. por año (T.Total).
VC_SV_ProfitByMonth_OverallTime	svProfitByMonth_OverallTime	Ganancia por mes (Tiempo total).
VC_SV_ProfitByWeek_OverallTime	svProfitByWeek_OverallTime	Ganancia por semana (Tiempo total).
VC_SV_ProfitByDay_OverallTime	svProfitByDay_OverallTime	Ganancia por día (Tiempo total).
VC_SV_SharpeRatio	svSharpeRatio	Sharpe Ratio.
VC_SV_Volatility	svVolatility	Volatilidad.
VC_SV_EstimatedAccountSize	svEstimatedAccountSize	Tamaño de cuenta.
VC_SV_SerieOfLosses	svSerieOfLosses	Serie de pérdidas.
VC_SV_WorstSerieOfLosses	svWorstSerieOfLosses	Peor serie de pérdidas.
VC_SV_SerieOfProfits	svSerieOfProfits	Serie de ganancias.
VC_SV_BestSerieOfProfits	svBestSerieOfProfits	Mejor serie de ganancias.
VC_SV_Open_EntryPrice	svOpen_EntryPrice	Punto de entrada (Pos. abiertas).
VC_SV_Open_ExitPrice	svOpen_ExitPrice	Punto de salida (Pos. abiertas).
VC_SV_Open_EntryDate	svOpen_EntryDate	Fecha de entrada (Pos. abiertas).
VC_SV_Open_ExitDate	svOpen_ExitDate	Fecha de salida (Pos. abiertas).
VC_SV_Open_EntryBar	svOpen_EntryBar	Barra de entrada (Pos. abiertas).
VC_SV_Open_ExitBar	svOpen_ExitBar	Barra de salida (Pos. abiertas).
VC_SV_Ratio	svRatio	Ratio: Ganancia anualizada/Peor serie de pérdidas.

EnumVCDSStatisticMeasurementUnit. Define las unidades en que se devuelven los resultados de la estadística de un sistema.

Valor	Descripción
VCDS_SMU_Undef	No definido.
VCDS_SMU_Money	En moneda (euro, dólar…).
VCDS_SMU_Points	En puntos.

VCDS_SMU_Perc	Porcentual.

EnumVCDSStatisticCompression. Define la compresión que se aplica a las variables estadísticas.

Valor	Descripción
VCDS_SC_Undef	No definido.
VCDS_SC_Trades	Compresión por negocios.
VCDS_SC_Secs	Compresión por segundos.
VCDS_SC_Mins	Compresión por minutos.
VCDS_SC_Days	Compresión por días.
VCDS_SC_Weeks	Compresión por semanas.
VCDS_SC_Months	Compresión por meses.
VCDS_SC_Years	Compresión por años.

EnumVCDSMarketPosition. Indica la posición que se tiene en el mercado.

Valor	Descripción
VCDS_P_Bull	Comprado.
VCDS_P_Bear	Vendido.
VCDS_P_Neutral	Neutro.

ENUMERACIONES COMTraderInterfaces

Se indican a continuación las enumeraciones de un tipo de variable de la librería **COMTraderInterfaces.**

EnumVCTOrderType. Contiene las definiciones de todos los tipos de órdenes disponibles en Visual Chart, y que se podrán utilizar en distintos objetos, métodos y eventos:

Propiedad	Descripción
VCT_OT_Limit	Tipo de orden limitada.
VCT_OT_StopLimit	Tipo de orden stop limitado.
VCT_OT_Market	Tipo de orden a mercado.
VCT_OT_StopMarket	Tipo de orden stop a mercado.
VCT_OT_StopMarketSimulated	Tipo de orden stop a mercado simulado.
VCT_OT_StopLimitSimulated	Tipo de orden stop limitado simulado.
VCT_OT_Best	Tipo de orden por lo mejor.
VCT_OT_BestStop	Tipo de orden stop por lo mejor.
VCT_OT_BestStopSimulated	Tipo de orden stop por lo mejor simulado.
VCT_OT_Unknown	Desconocido (no debe usarse nunca).

EnumVCTVolumeRestriction. Contiene las definiciones de los distintos valores que puede tomar una restricción de volumen para una orden:

Propiedad	Descripción
VCT_VR_NoRestriction	Sin restricción.
VCT_VR_AllNothing	Restricción Todo o Nada.
VCT_VR_Hide	Restricción Volumen oculto. Debe indicarse el volumen oculto en la propiedad HideVolume de VCT_Order .
VCT_VR_MinVolume	Restricción Volumen mínimo. Debe indicarse el volumen mínimo en la propiedad

	MinVolume de VCT_Order .
VCT_VR_Unknown	Desconocido (no debe usarse nunca).

EnumVCTTimeRestriction. Contiene las definiciones de los distintos valores que puede tomar una restricción de tiempo para una orden:

Propiedad	Descripción
VCT_TR_NoRestriction	Sin restricción.
VCT_TR_Immediate	Restricción Inmediata.
VCT_TR_OpenAuction	Restricción Subasta de Apertura.
VCT_TR_CloseAuction	Restricción Subasta de Cierre.
VCT_TR_Auction	Restricción Cualquier Subasta.
VCT_TR_Session	Restricción Sesión Actual.
VCT_TR_Date	Restricción Hasta una fecha (Fecha de Validez). Debe indicarse la fecha en la propiedad ValidDate de VCT_Order .
VCT_TR_Unknown	Desconocido (no debe usarse nunca).

EnumVCTOrderSide. Contiene las definiciones de los signos (compra o venta) que caracterizan a una orden, posición abierta y operación cerrada:

Propiedad	Descripción
VCT_OS_Buy	Compra.
VCT_OS_Sell	Venta.
VCT_OS_Unknown	Desconocido (no debe usarse nunca).

EnumVCTOrderStatus. Contiene las definiciones de los distintos estados que puede tomar una orden:

Propiedad	Descripción
VCT_OST_WaitingMarket	Enviando.
VCT_OST_PendingEvent	Pendiente de evento (por una condición propia o por el enlace con otra orden).
VCT_OST_OnSimulation	En simulación (para los tipos de órdenes simulados).
VCT_OST_Market	En mercado.
VCT_OST_Canceled	Cancelada.
VCT_OST_PendingCancelation	Pendiente de cancelación.
VCT_OST_Filled	Ejecutada.
VCT_OST_PendingModification	Pendiente de modificación.
VCT_OST_PartialFilled	Parcialmente ejecutada.
VCT_OST_Rejected	Rechazada.
VCT_OST_Pending24x7	Pendiente de apertura de mercado (esperando en sistema 24x7).
VCT_OST_StopPending	Pendiente de disparo de stop.
VCT_OST_Unknown	Desconocido (no debe usarse nunca).

EnumVCTOrderFilter. Contiene las definiciones de los distintos valores que puede tomar un filtro a la hora de realizar una consulta de órdenes. En el momento de realizar una consulta, se puede establecer el filtro con varios de los valores siguientes, a través de una operación OR (una suma). Por ejemplo, para consultar las órdenes activas y ejecutadas, se debe establecer el filtro como VCT_OF_Active + VCT_OF_Filled.

Propiedad	Descripción
VCT_OF_All	Consulta de todas las órdenes.
VCT_OF_Active	Consulta de órdenes activas.
VCT_OF_Filled	Consulta de órdenes ejecutadas.
VCT_OF_Canceled	Consulta de órdenes canceladas.
VCT_OF_PartialExe	Consulta de órdenes parcialmente ejecutadas.
VCT_OF_PartialExeCancelled	Consulta de órdenes parcialmente ejecutadas y canceladas.
VCT_OF_Market	Consulta de órdenes en mercado.

EnumVCTOrderLocation. Contiene las posibles localizaciones de una orden.

Propiedad	Descripción
VCT_OL_Local	Local
VCT_OL_Server	Servidor
VCT_OL_Bridge	Puente/Enlace
VCT_OL_Market	Mercado

EnumVCTSource. Contiene las definiciones de los distintos orígenes que pueden generar la orden. Una orden puede generarse desde un sistema, una tabla, a través del servidor COM, etc.

Propiedad	Descripción
VCT_SRC_Unknown	Desconocido.
VCT_SRC_Systems	Sistemas.
VCT_SRC_Tables	Tablas.
VCT_SRC_Positions	Posiciones.
VCT_SRC_GraphicObject	Objeto gráfico.
VCT_SRC_Charts	Gráfico.
VCT_SRC_Alerts	Alerta.
VCT_SRC_Phone	Teléfono.
VCT_SRC_Tv	Televisión.
VCT_SRC_Web	Web.
VCT_SRC_PDA	PDA.
VCT_SRC_CellPhone	Teléfono móvil.
VCT_SRC_email	Correo electrónico.
VCT_SRC_External	Externa.
VCT_SRC_COM	Desde el servidor COM.

EnumVCTOpenPositionStatus. Indica el estado de una posición abierta.

Propiedad	Descripción
VCT_OPS_NonTrade	Posición no negociable.
VCT_OPS_TradeGrouped	Posición negociable y agrupable con otras posiciones.
VCT_OPS_TradeNoGrouped	Posición negociable no agrupable con otras posiciones.
VCT_OPS_NonTradePledge	Posición pignorada, no negociable.

EnumVCTRelatedType. Indica la relación existente entre distintas órdenes. Las de tipo OCO, OSO y Bracket generan varias órdenes relacionadas entre sí. (Ver RelatedId y RelatedType de VCT_Order).

Propiedad	Descripción
VCT_RT_NoRelation	Sin relación con otra orden.
VCT_RT_OSOMainOrder	Indica que la orden es la principal de una OSO. RelatedId contiene el identificador de la propia orden.
VCT_RT_OSOWaitingOrder	La orden se envía cuando se ejecute su orden principal. RelatedId contiene el identificador de la orden principal.
VCT_RT_OCOOrder	La orden se cancela si se ejecuta la orden cuyo identificador coincida con el valor de RelatedId .
VCT_RT_BracketMainOrder	Indica que la orden es la principal de una Bracket. RelatedId contiene el identificador de la propia orden.
VCT_RT_BracketStopOrder	La orden es el stop de la Bracket. RelatedId contiene el identificador de la orden principal.
VCT_RT_BracketLimitOrder	La orden es la limitada de la Bracket. RelatedId contiene el identificador de la orden principal.

ENUMERACIONES VCContributor

Se indican a continuación los distintos valores que puede tomar un tipo de variable concreta de la librería **VCContributor**.

EnumVCCAlertPriority. Indica la prioridad que tiene la alerta. Las de mayor prioridad se muestran durante más tiempo.

Propiedad	Descripción
VCC_AP_Inmediate	Inmediata (VisualChart la muestra inmediatamente, en cuanto llega).
VCC_AP_High	Prioridad alta.
VCC_AP_Medium	Prioridad media.
VCC_AP_Low	Prioridad baja.
VCC_AP_Default	Prioridad por defecto.

EnumVCCtbType. Las contribuciones pueden clasificarse en distintos grupos. El tipo de una contribución indica en qué grupo está.

Propiedad	Descripción
VCC_CT_None	Ninguno.
VCC_CT_Indicators	Indicadores.
VCC_CT_Charting	Gráficos.

VisualChart

Real Time Financial Information & Trading Software