
# Visual Chart 6 - Funciones y propiedades

## Guía rápida para el usuario

visualchart

# Indicadores

Alert
Angle
CalcDate
CalcTime
Close
CurrentBar
Date
DateSubstract
GetFeedFields
GetBackgroundColor
GetBarColor
GetBarStyle
GetBarRepresentation
GetBarWidth
GetExitOrder
GetHighest
GetHighestBar
GetHistogramBand
GetIndicatorIdentifier GII
GetIndicatorPos
GetIndicatorValue GIV
GetLineName
GetLowest
GetLowestBar
GetNthHighest
GetNthLowest
GetOrderCount
GetOrderDate
GetOrderLabel
GetOrderPrice
GetOrderSide
GetOrderSymbolCode
GetOrderType
GetOrderVolume
GetPivotDown
GetPrice
GetSwingHigh
GetSwingHighBar
GetSwingLow
GetSwingLowBar
GetSymbolIdentifier GSI
GetSymbolInfo
GetSymbolInfoEx
GetSystemIdentifier GSYSI
GetTrueHigh
GetTrueLow
GetTrueRange
GetTrueRangeCustom
GetVolatility
GetWndBackGroundColor
LimitOrder
LimitPrice
LimitVol
Low
MinutesToTime
NumberOfLines
Open
OpInt
ReleaseDataIdentifier RDI
RegressionAngle
RegressionSlope
SetBackgroundColor
SetBarColor
SetBarProperties
SetBarRepresentation
SetBarStyle
SetBarWidth
SetHistogramBand
SetIndicatorPos
SetIndicatorValue
SetLineName
SetWndBackgroundColor
ShouldTerminated
Slope
StarBar
Time
TimeEx
TimeToMinutes
Volume

# Estrategias

Angle
AvgBarsInTrade
AvgLosingTrade
AvgTrade
AvgWinningTrade
BestSeries
Buy
CalcDate
CalcTime
Close
ConfigStk
CurrentBar
CurrentContract
CurrentEntries
Date
DateSubstract
EndOfDayTime
ExitLong
GetExitPrice
GetExitTime
GetFeedFields
GetHighest
GetHighestBar
GetIndicatorIdentifier GII
GetIndicatorValue GIV
GetLowest
GetLowestBar
GetMarketPosition
GetMaxContracts
GetMaxEntries
GetNthHighest
GetNthLowest
GetOrderCount
GetOrderCount
GetOrderDate
GetOrderLabel
GetSwingLow
GetSwingLowBar
GetSymbolIdentifier GSI
GetSymbolInfo
GetSymbolInfoEx
GetSystemIdentifier GSYSI
GetTrueHigh
GetTrueLow
GetTrueRange
GetTrueRangeCustom
GetVolatility
GrossLoss
GrossProfit
High
IsFirstDayBar
IsLastDayBar
LargestLosingTrade
LargestWinningTrade
ReleaseDataIdentifier RDI
RegressionAngle
RegressionSlope
Sell
ShouldTerminated
Slope
StandardDeviation
StarBar
Time
TimeEx
TimeToMinutes
TodayCurrentBar
TodayHigh
TodayLow
Volume
WorstSeries

ExitPositionsAtEndOfDay
ExitShort
ExitTrailingLimit
ExitTrailingStop
FilledOrders
GetBarsSinceEntry
GetBarsSinceExits
GetConfigStk
GetDailyLosers
GetDailyWinners
GetEntryDate
GetEntryPrice
GetEntryTime
GetExitDate
GetExitOrder
GetOrderPrice
GetOrderSide
GetOrderSymbolCode
GetOrderType
GetOrderVolume
GetOrderVolume
GetPivotDown
GetPivotUp
GetPositionProfit
GetPrice
GetStkLength
GetStkValue
GetStkValues
GetSwingHigh
GetSwingHighBar
LC_Index
LimitOrder
LimitPrice
LimitVol
Low
MarketFilledOrders
MinutesToTime
NetProfit
NumberOfLosingTrades
NumberOfTrades
NumberOfWinningTrades
Open
OpInt
PercentProfitable
ProfitFactor

# Estudios

Angle
CalcDate
CalcTime
Close
CurrentBar
Date
DateSubstract
FeedFields
GetExitOrder
GetHighest
GetHighestBar
GetIndicatorIdentifier GII
GetIndicatorValue GIV
GetLowest
GetLowestBar
GetNthHighest
GetNthLowest
GetOrderCount
GetOrderDate
GetOrderLabel
GetOrderPrice
GetOrderSide
GetOrderSymbolCode
GetOrderType
GetOrderVolume
GetPivotDown
GetPivotUp
GetPrice
GetSwingHigh
GetSwingHighBar
GetSwingLow
GetSwingLowBar
GetSymbolIdentifier GSI
GetTrueHigh
GetTrueLow
GetTrueRange
GetTrueRangeCustom
GetVolatility
High
LimitOrder
LimitPrice
LimitVol
Low
MinutesToTime
Open
OpInt
PaintBar
PaintCandlestick
PaintMaxMin
PaintSeries
ReleaseDataIdentifier
RDI
RegressionAngle
RegressionSlope
ShouldTerminated
Slope
StarBar
Time
TimeEx
TimeToMinutes
Volume

## Alert

**Descripción:**
Esta función se utiliza en la programación de indicadores para emitir alertas. Al cumplirse unas determinadas condiciones, se mostrará un aviso en pantalla con el mensaje deseado.

**Sintaxis:**
Alert(Description)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Description | "Indicator Alert" | Texto que se mostrará cuando se dispare la alerta. |

Para que se visualice el mensaje de alerta en pantalla, es preciso que una vez que se haya insertado el indicador, se active la propiedad **Alertas Indicador** en el editor de propiedades de este.

**Ejemplo (VB.NET):**

```
Me.Alert("Atención, Cruce de Medias")
```

Mostrará en pantalla el mensaje "Atención, Cruce de Medias"

## Angle

**Descripción:**
Devuelve el valor del ángulo que forma con la horizontal, la recta de regresión formada por los precios StartPrice y EndPrice, que relaciona las cotizaciones con la variable de tiempo.

**Sintaxis:**
Me.Identifier.Angle(StartBar, EndBar, StartPrice, EndPrice)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| StarBar | - | Número de barra de inicio de la recta. |
| EndBar | - | Número de barra de fin de la recta. |
| StarPrice | - | Precio de inicio de la recta. |
| EndPrice | - | Precio de fin de la recta. |

**Ejemplo (VB.NET):**

Supongamos que deseamos conocer, para cada momento, el valor en radianes del ángulo entre el precio de cierre actual y el del cierre de hace 30 barras. En primer lugar tendríamos que definir la variable de salida.

```
Dim anguloenradianes As Double  = Me.Data.Angle(Bar-30,Bar,Me.Data.Close(30),Me.Data.Close(0))
```

A esta variable se le asignará el valor que devuelva la llamada a esta propiedad.

## AvgBarsInTrade

**Descripción:**
Devuelve el promedio de número de barras existentes durante el periodo de vida de los negocios. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
AvgBarsInTrade

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Supongamos que deseamos saber la media de barras por negocio. En primer lugar, se define una variable de salida:

Dim mediabarsneg as Double

Se calcula la media de barras por negocio cada vez que se genere uno nuevo, haciendo la siguiente pregunta:

If NumberOfTrades > 1 then

Con esto nos aseguramos de que al menos se han generado dos negocios

mediabarsneg = Me.AvgBarsInTrade

Se asignará a la variable de salida el valor que devuelve la propiedad

Luego se puede almacenar en otra variable previamente definida, el número de negocios que llevamos:

Dim numnegactuales As Long = Me.NumberOfTrades

Volveríamos a calcular la media de barras por negocio solo cuando *NumberOfTrades* > NumNegActuales

## AvgLosingTrade

**Descripción:**
Devuelve el promedio de resultados de los negocios perdedores. Este valor irá cambiando a medida que se generen nuevas barras y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

El valor devuelto es del tipo *SttRepresentation*.

**Sintaxis:**
AvgLosingTrade(Show)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

```
Dim gananciamediaperdedores As Double = Me.AvgLosingTrade(SttRepresentation.ByPoints)
```

Asigna a la variable la ganancia media de todos los negocios perdedores en puntos.

## AvgTrade

**Descripción:**
Devuelve el promedio de resultados de los negocios. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

El valor devuelto es del tipo *SttRepresentation*.

**Sintaxis:**
AvgTrade(Show)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

```
Dim gananciamedia = Me.AvgTrade(SttRepresentation.Porcentual)
```

Asigna a la variable la ganancia media de todos los negocios en tanto por ciento.

## AvgWinningTrade

**Descripción:**
Devuelve el promedio de resultados de los negocios ganadores. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a esta propiedad.

El valor devuelto es del tipo *SttRepresentation*.

**Sintaxis:**
AvgWinningTrade(Show)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Dim gananciamediaganadores = Me.AvgWinnigTrade(SttRepresentation.ByPoints)

Asigna a la variable GananciaMediaGanadores (previamente definida) la ganancia media de todos los negocios positivos (en puntos).

## BestSeries

**Descripción:**
Devuelve el mejor valor de ganancia total que se ha llegado a alcanzar. Este irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a esta propiedad.

**Sintaxis:**
BestSeries(Show)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Me.BestSeries(Porcentual)

Retornará en tanto por ciento, el mejor valor de ganancia total que se ha alcanzado hasta el momento de aplicar esta propiedad.

## Buy

**Descripción:**
Esta función es utilizada para dar órdenes de compra. Es usa indistintamente para comprar acciones, futuros, etc.

**Sintaxis:**
Buy(TraderType, Contracts, Price, Label)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|

| | | |
|---|---|---|
| Type | TraderType.AtStop | Tipo de orden que se desea lanzar (TraderType.AtClose, TraderType.AtMarket, TraderType.AtLimit y TraderType.AtStop). |
| Contracts | 1 | Número de contratos/acciones. Las especificaciones numéricas sobre contratos pueden ser sustituidas por variables o por cualquier función definida previamente. |
| Price | - | Precio de compra. Sólo se indica este parámetro en las órdenes del tipo TraderType.AtStop y TraderType.AtLimit. El valor puede expresarse mediante un número, una variable, una función, o bien una mezcla de valor y función. |
| Label | - | Etiqueta de la orden en formato texto. |

**Ejemplo (VB.NET):**

```
If (Me.GetMarketPosition() <> 1) then
    Me.Buy(TraderType.AtStop, 1, Me.Data.High() +10, "C1")
End If
```

Lanza una orden de compra en stop de un contrato, siendo el precio del stop el máximo de la barra más 10 puntos, y la etiqueta identificativa de la orden “C1”.

## CalcDate

**Descripción:**
Suma una cantidad de días a un día en cuestión y devuelve la fecha resultante en **formato militar (AAAAMMDD).**

**Sintaxis:**
CalcDate(Date, Days)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Date | - | Fecha (AAAAMMDD) a la cual se le sumará una cantidad de días. |
| Days | - | Cantidad de días a sumar. |

**Ejemplo (VB.NET):**

```
Dim newdate As Long = Me.CalcDate(2015110,5)
```

Suma 5 días a la fecha 10/01/2015, que en formato militar es 20150110, por lo tanto retorna 20150115, que en formato de fecha es 15/01/2015.

## CalcTime

**Descripción:**
Suma una cantidad de minutos a una hora en cuestión, y devuelve la hora resultante en formato militar (HHMM).

**Sintaxis:**
CalcTime(Time, Minutes)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Time | - | Hora (HHMM) a la cual se le sumará una cantidad de minutos. |
| Minutes | - | Cantidad de minutos a sumar. |

**Ejemplo (VB.NET):**

```
Dim newtime As Long = Me.CalcTime(1000,30)
```

Suma 30 minutos a las 10:00 am (formato militar 1000), por lo tanto retorna 1030, que en formato de hora es 10:30 am.

## Close

**Descripción:**
Esta función devuelve el valor de cierre de una barra. La propiedad *Close* la podemos encontrar en cualquier serie de datos o indicador.

**Sintaxis:**
Identifier.Close(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hace referencia a la barra actual. En este parámetro es posible indicar un valor numérico contenido en una variable o teclear directamente el valor.<br>Puede especificarse como función sustituyendo el valor numérico.<br>Este parámetro sólo permite valores positivos. |

**Ejemplo (VB.NET):**

```
Me.Data.Close(3,Data1)
```

Devuelve el cierre de hace 3 barras de la fuente de datos principal.

## ConfigStk

**Descripción:**
Permite establecer las propiedades iniciales de las estadísticas que se quieran extraer.

**Sintaxis:**
ConfigStk(Sing, Unit, Filt, CompType, Compression, BeginDate, EndDate)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| sign | StatisticSign.Nets | Filtro de resultados según sean los negocios ganadores (StatisticSign.Wins), perdedores (StatisticSign.Loss) o todos (StatisticSign.Nets). |
| unit | StatisticUnit.Money | Representación de los datos en moneda (suMoney), en puntos (suPnts) o en porcentaje (ssPorc). |
| filt | StatisticFilt.All | Filtro de resultados por posición, es decir, según sean largos (sfLong) o cortos (sfShort). |
| compType | StatisticCompType.Trades | Representación de los datos agrupados por negocios (sctTraders), días (sctDays), semanas (sctWeeks), meses (sctMoths) o años(sctYears). |
| compression | 1 | Unidad de compresión. |
| beginDate | Nothing | Fecha de inicio del intervalo de tiempo durante el cual se quieren extraer datos. |
| endDate | Nothing | Fecha fin del intervalo de tiempo durante el cual se quieren extraer datos. |

**Ejemplo (VB.NET):**

```
Me.ConfigStk(StatisticSign.Loss, StatisticUnit.Money, StatisticFilt.Short, StatisticCompType.Days,1,
cDate("09/08/2010"), cDate("09/08/2015")
```

Establece las propiedades de la estadística de una estrategia, para obtener la información en moneda, de la pérdida por día (sólo aquellos en los que se ha entrado a corto), desde el 9 de agosto del 2010 al 9 de septiembre del 2015.

## CurrentBar

**Descripción:**
Esta función retorna el número de orden de la barra en la que se están realizando los cálculos (barra actual). Considerando la primera barra de la serie de datos como la barra número 0, se irá incrementando en una unidad cada barra evaluada.

Cuando se trabaja con dos series de datos y una tiene más histórico que otra, los cálculos comenzarán cuando una barra de la primera serie coincida en el tiempo con una barra de la segunda serie. Esta barra será considerada como primera barra (CurrentBar=1).

**Sintaxis:**
CurrentBar

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

```
Me.CurrentBar()
```

Suponiendo que los cálculos se están efectuando en la séptima barra de la serie de datos, esta función retornaría el valor 6.

## CurrentContract

**Descripción:**
Esta función se utiliza para conocer el número de contratos o de acciones que hay comprados o vendidos en la posición que está abierta.

**Sintaxis:**
CurrentContract

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Me.CurrentContract En el caso de que no haya, la función devuelve el valor 0.

## CurrentEntries

**Descripción:**
Esta función se utiliza para conocer el número de entradas diferentes que hay abiertas en una posición. Una posición puede tener varias entradas diferentes en función de la etiqueta establecida en la orden y de la modalidad de casar operaciones elegida.

**Sintaxis:**
CurrentEntries

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Me.CurrentEntries

La función devuelve el número de entradas que haya en vigor en el momento de aplicar la función. Si no hay ninguna, retorna el valor 0.

## Date

**Descripción:**
Todas las barras de un gráfico tienen una fecha, tanto si se trata de un gráfico de fin de día como si se trata de un gráfico intradiario. Permite obtener de cada barra la fecha en que se produjo. El formato en el que se retorna la fecha es militar (**AAAAMMDD)**. Por tanto, si la barra se produjo el día 10 de agosto de 2000, esta función retornará el valor numérico de 20000810.

La propiedad *Date* la podemos encontrar en cualquier serie de datos o indicador.

**Sintaxis:**
Identifier.Date(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hace referencia a la barra actual. En este parámetro es posible indicar un valor numérico contenido en una variable o teclear directamente el valor.<br>Puede especificarse como función sustituyendo el valor numérico.<br>Este parámetro sólo permite valores positivos. |

**Ejemplo (VB.NET):**

Me.Data.Date(3)

La función devuelve la fecha correspondiente a hace 3 barras de la serie de datos principal.

## DateSubstract

**Descripción:**
Devuelve la diferencia (en días) entre dos fechas**.**

**Sintaxis:**
DateSubstract(Left, Right)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Left | - | Minuendo (fecha, en formato militar, a la que se resta) |
| Right | - | Sustraendo (fecha, en formato militar, que resta) |

**Ejemplo (VB.NET):**

Dim diff As Long = Me.DateSubstract(20150110, 20150120)  La función retorna el valor 10.

## EndOfDayTime

**Descripción:**
Retorna la hora de cierre de la sesión actual  (de la fuente principal) en formato militar (HHMM).

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Sintaxis:**
Me.EndOfDayTime()

**Ejemplo (VB.NET):**

Queremos entrar a un precio determinado siempre y cuando estemos fuera de mercado y no sea la última barra de la sesión:

```
If GetMarketPosition() = 0 And Me.entryprice <> 0 Then
  If (Me.Data.Time <> Me.EndOfDayTime()) Then
    Me.Buy(TradeType.AtStop, 2, Me.entryprice * (1 + Me.factor / 100))
    Me.Sell(TradeType.AtStop, 2, Me.entryprice * (1 - Me.factor / 100))
  End If
End If
```

Comparamos la propiedad *Time()* con la propiedad *EndOfDayTime()* para comprobar que son distintas.

## ExitLong

**Descripción:**
Esta función se utiliza cuando deseamos cerrar una posición de compra y no posicionarnos a corto (o hacer una venta a crédito). Por ejemplo, si se da una orden de compra de 500 acciones, y se cumplen las condiciones que se han establecido para salir de ese negocio, se utilizará esta función.

**Sintaxis:**
ExitLong(TraderType, Contracts, Price, Label)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Type | TraderType.AtStop | Tipo de orden que se desea lanzar (TraderType.AtClose, TraderType.AtMarket, TraderType.AtLimit y TraderType.AtStop). |
| Contracts | 1 | Número de contratos/acciones. Las especificaciones numéricas sobre contratos pueden ser sustituidas por variables o por cualquier función definida previamente. |
| Price | - | Precio de compra. Sólo se indica este parámetro en las órdenes del tipo TraderType.AtStop y TraderType.AtLimit. El valor puede expresarse mediante un número, una variable, una función, o bien una mezcla de valor y función. |
| Label | - | Etiqueta de la orden en formato texto. |

- Si no se especifica el número de acciones/contratos (Contracts) ni la etiqueta (Label, suponiendo que se están usando varias órdenes de compra), se cerrará la posición entera y saldremos del mercado.
- Si no se especifica el número de acciones pero sí la etiqueta, en el supuesto de que estemos usando varias órdenes de compra, se cerrarán todos los contratos/acciones correspondientes a la orden que tenía esa etiqueta.
- Si se especifica el número de contratos y la etiqueta, se cerrará ese número de contratos/acciones de la orden cuya etiqueta se ha indicado.

- Si se especifica el número de contratos/acciones pero no se especifica la etiqueta, suponiendo que hay varias posiciones abiertas, se cerrarán los contratos/acciones especificados en la última orden.

**Ejemplo (VB.NET):**

Me.ExitLong(TraderType.AtStop, 1, Me.GetEntryPrice -10, "C1")

Se cerrará la posición (1 contrato), de la orden etiquetada como “C1”, con una venta en stop al precio de entrada menos 10 puntos. En este caso se trata de limitar las pérdidas con un stop de protección.

Me.ExitLong(TraderType.AtLimit, 1, Me.GetEntryPrice+30)

Se cerrará la posición (1 contrato) con una venta limitada al precio de entrada más 30 puntos. En este caso el objetivo es cerrar por beneficios.

## ExitPositionsAtEndOfDay

**Descripción:**
Si se establece la propiedad a *True*, entonces todos los negocios de la estrategia se liquidarán al cierre de cada sesión.

Atención. Esta posibilidad se admite para todos los casos de desarrollos de estrategias en backtesting, si embargo, si se desea activar el *trading* de una estrategia en tiempo real que lo utilice, es necesario contactar primero con el bróker del usuario para confirmar si el método está contemplado dentro de la metodología de dicho bróker.

En caso de que queramos hacer uso de ésta propiedad, se recomienda declararla desde el método *OnInitCalculate().*

**Sintaxis:**
Me.ExitPositionsAtEndOfDay() = True/False

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Queremos que cierre todos los negocios al cierre de sesión en base a un parámetro. Primero, declaramos el parámetro:

```
<Parameter(Name:="EndOfDay", DefaultValue:=1, MinValue:=0, MaxValue:=1, Step:=1)>
Private endofday As Integer
```

Seguidamente, establecemos la propiedad a verdadero o falso en base al parámetro:

```
Public Overrides Sub OnInitCalculate()
   Me.ExitPositionsAtEndOfDay = (Me.endofday = 1)
End Sub
```

## ExitShort

**Descripción:**
Se utiliza cuando se quiere cerrar una posición de venta a crédito y no posicionarse a largo (o hacer una compra). Por ejemplo, si se da una orden de venta de 5 contratos de un futuro, y se cumplen las condiciones para liquidar la posición, será necesario usar esta función para cerrar dicha posición.

**Sintaxis:**
ExitShort(TraderType, Contracts, Price, Label)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Type | TraderType.AtStop | Tipo de orden que se desea lanzar (TraderType.AtClose, TraderType.AtMarket, TraderType.AtLimit y TraderType.AtStop). |
| Contracts | 1 | Número de contratos/acciones. Las especificaciones numéricas sobre contratos pueden ser sustituidas por variables o por cualquier función definida previamente. |
| Price | - | Precio de compra. Sólo se indica este parámetro en las órdenes del tipo TraderType.AtStop y TraderType.AtLimit. El valor puede expresarse mediante un número, una variable, una función, o bien una mezcla de valor y función. |
| Label | - | Etiqueta de la orden en formato texto. |

- Si no se especifica el número de acciones/contratos (Contracts) ni la etiqueta (Label, en el supuesto de que se estén usando varias órdenes de venta), se cerrará la posición entera y saldremos del mercado.
- Si no se especifica el número de acciones pero sí la etiqueta (supuesto de que se estén usando varias órdenes de venta), se cerrarán todos los contratos/acciones correspondientes a la orden que tiene esa etiqueta.
- Si se especifica el número de contratos y la etiqueta, entonces se cerrará ese número de contratos/acciones de la orden cuya etiqueta se ha especificado.
- Si se especifica el número de contratos/acciones pero no se especifica la etiqueta, si hay varias posiciones abiertas, se cerrarán los contratos especificados de la última orden.

**Ejemplo (VB.NET):**

```
Me.ExitShort(TraderType.AtStop, 1, Me.Data.High() +10, "ES1")
```

Se cerrará la posición (1 contrato), de la orden etiquetada como “ES1”, con una compra en stop al precio máximo de la barra más 10 puntos. En este caso se trata de limitar las pérdidas con un stop de protección.

## ExitTrailingLimit

**Descripción:**
Este método permite aplicar una orden *trailing limit* para un negocio abierto concreto. Esta función supone una novedad en Visual Chart 6 y facilita la labor en el diseño de órdenes de salida dinámicas.

La orden *trailing limit* consiste en lo siguiente (en caso de estar abiertos a largo):

- Almacena el valor más alto alcanzado desde que abrió la posición.
- Calcula el precio de salida en base a dicho máximo según la siguiente función:
  - limitPrice = Maximo + (Maximo x *percentage* x 0.01)
- Si está en el inicio del negocio, envía una orden *ExitLong* limitada al precio calculado.
- Para el resto de barras, si el valor resultante es **inferior** al último valor calculado, se sustituye y modifica el precio de la orden de salida.

**Sintaxis:**
Me.ExitTrailingLimit(percentage)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Percentage | - | Tanto por ciento de margen que se aplica al valor más extremo para posicionar la orden de salida limitada. |

### Ejemplo (VB.NET):

```
If GetMarketPosition() <> 0 Then
   Me.ExitTrailingLimit(pcttrail)
End If
```

Cada vez que tengamos una posición abierta, se activará el TrailingLimit con un porcentaje que dependerá del parámetro *pcttrail*.

## ExitTrailingStop

**Descripción:**
Este método permite aplicar una orden *trailing stop* para un negocio abierto concreto. Esta función supone una novedad en Visual Chart 6 y facilita la labor en el diseño de órdenes de salida dinámicas.

La orden *trailing stop* consiste en lo siguiente (en caso de estar abiertos a largo):

- Almacena el valor más alto alcanzado desde que abrió la posición.
- Calcula el precio de salida en base a dicho máximo según la siguiente función:
  - stopPrice = Maximo - (Maximo x *percentage* x 0.01)
- Si está en el inicio del negocio, envía una orden *ExitLong* en stop al precio calculado.
- Para el resto de barras, si el valor resultante es **superior** al último valor calculado, se sustituye y modifica el precio de la orden de salida.

**Sintaxis:**
Me.ExitTrailingStop(percentage)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Percentage | - | Tanto por ciento de margen que se aplica al valor más extremo para posicionar la orden de salida en stop. |

### Ejemplo (VB.NET):

```
If GetMarketPosition() <> 0 Then
   Me.ExitTrailingStop(pcttrail)
End If
```

Cada vez que tengamos una posición abierta, se activará el TrailingStop con un porcentaje que dependerá del parámetro *pcttrail*.

## FilledOrders

**Descripción:**
La propiedad FilledOrders permite conocer si en una barra se han ejecutado órdenes activas de compra o venta asociadas a una etiqueta en concreto. Si se han ejecutado, devolverá un 1, en otro caso devolverá un 0.

**Sintaxis:**
FilledOrders(Label, Side, BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Label | - | Etiqueta asociada a la orden que se desea consultar. |
| Side | - | Posición de la orden a consultar. (0 posición de compra y 1 posición de venta) |
| BarsAgo | 0 | Barra de la que se desea extraer la fecha. El valor por defecto es la barra actual. |

**Ejemplo 1 (VB.NET):**

Me.FilledOrders("C1", 0, 10)

En el caso de que hace 10 barras se haya ejecutado una orden de compra, etiquetada como "C1", devolverá el valor 1.

**Ejemplo 2 (VB.NET):**

En una estrategia, lanzamos tres órdenes en stop, dos de compra con las etiquetas A y B y una de venta, con la etiqueta C.

En la barra siguiente, queremos saber cuántas de las tres órdenes se han ejecutado.

Para ello hacemos lo siguiente:

```
Definimos un contador de órdenes de compra y otro de venta.
Dim contadorcompras As Integer = 0
Dim contadorventas As Integer = 0
If Me.FilledOrders("A",0 , 0) =1 then
    contadorcompras= 1
End If
If Me.FilledOrders("B",0 , 0) = 1 then
    contadorcompras=contadorcompras+1
End If
```

```
If Me.FilledOrders("C",1 , 0) =1 then
    contadorventas= 1
End If
```

## GetBackGroundColor

**Descripción:**
Retorna el color de fondo de la ventana de una barra determinada.

**Sintaxis:**
GetBackGrounColor (BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

Me.GetBackGroundColor(0) Devuelve el color de fondo de la ventana para la barra actual.

## GetBarColor

**Descripción:**
Retorna el color de la línea de datos indicada en una barra determinada.

**Sintaxis:**
GetBarColor (BarsAgo, Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | - | Línea, cuyo color en la barra indicada, se desea obtener. |

**Ejemplo (VB.NET):**

Me.GetBarColor(1,3) Devuelve el color de la barra anterior de la línea 3.

## GetBarsSinceEntry

**Descripción:**
Esta función se utiliza para conocer el número de barras que se han completado desde que se abrió una posición determinada (de compra o de venta). Si no se ha abierto ninguna posición retorna el valor 0.

**Sintaxis:**
GetBarsSinceEntry(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de negocios hacia atrás. El valor por defecto hace referencia al negocio actual. |

**Ejemplo (VB.NET):**

Me.GetBarsSinceEntry (0)  Retorna el número de barras formadas desde que se abrió la posición actual.

## GetBarsSinceExit

**Descripción:**
Esta función se utiliza para conocer el número de barras que se han completado desde que se cerró una posición determinada. Si no se ha salido en ninguna posición retorna el valor 0.

**Sintaxis:**
GetBarsSinceExit(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de negocios hacia atrás. El valor por defecto hace referencia al negocio actual. |

**Ejemplo (VB.NET):**

Me.GetBarsSinceExit (1)  Retorna el número de barras formadas desde cierre del negocio anterior.

## GetBarStyle

**Descripción:**
Esta función devuelve el estilo de la línea que se utiliza en una barra determinada. El valor devuelto es del tipo *LineStyle*.

**Sintaxis:**
GetBarStyle(BarsAgo, Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0  hace referencia a la barra actual. |
| Line | - | Identifica la línea a la que pertenece la barra de la que se quiere obtener el estilo de línea. |

**Ejemplo (VB.NET):**

Me.GetBarStyle (0,1)

Retorna el estilo de línea en la barra actual de la línea 1. Los valores que puede devolver la función son:

- **LineStyle.Solid:** Línea contínua.
- **LineStyle.Dash:** Línea discontinua.
- **LineStyle.Dot:** Línea punteada.
- **LineStyle.DashDot:** Línea discontinua con punto.
- **LineStyle.DashDotDot:** Línea discontinua con dos puntos.

## GetBarRepresentation

**Descripción:**
Esta función devuelve el tipo de representación que se está aplicando en el indicador para una línea en cierta barra. El valor devuelto es del tipo *IndicatorRepresentation*.

**Sintaxis:**
GetBarsSinceExit(BarsAgo, Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | - | Identifica la línea a la que pertenece la barra de la que se quiere obtener el estilo de línea. |

**Ejemplo (VB.NET):**

```
If Me.GetBarRepresentation(10, 1) = IndicatorRepresentation.Lineal The
      MsgBox("La línea 1 se representó con una línea hace 10 barras.")
End If
```

Los valores que puede devolver la función son:

- **IndicatorRepresentation.Bars:** Representación por barras.
- **IndicatorRepresentation.Candlestic:** Representación por velas.
- **IndicatorRepresentation.DottedLine:** Representación por línea discontínua.
- **IndicatorRepresentation.FilledHistogram:** Representación por histograma relleno.
- **IndicatorRepresentation.Histogram:** Representación por histograma.
- **IndicatorRepresentation.Lineal:** Línea contínua.
- **IndicatorRepresentation.Parabolic:** Representación por líneas parabólicas.
- **IndicatorRepresentation.Lineal:** Línea punteada.
- **IndicatorRepresentation.Volume:** Representación del tipo volumen.

## GetBarWidth

**Descripción:**
Retorna el grosor de una línea concreta para la barra indicada.

**Sintaxis:**
GetBarWidth(BarsAgo, Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | | Identifica la línea de datos a la que pertenece la barra de la que se desea conocer el grosor. |

**Ejemplo (VB.NET):**

Me.GetBarWidth(0,1)  La función devuelve el grosor(1,2,…) de la línea 1 en la barra actual.

## GetConfigStk

**Descripción:**
Este procedimiento retorna las propiedades iniciales de las estadísticas que en ese momento se estén utilizando.

**Sintaxis:**
GetConfigStk(Sing, Unit, Filt, CompType,Compression, BeginDate, EndDate)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Sing | - | Filtro de resultados según sean los negocios ganadores (StatisticSign.Wins) o perdedores (StatisticSign.Loss). |
| Unit | - | Representación de los datos en moneda (StatisticUnit.Money), en puntos (StatisticUnit.Pnts) o en porcentaje (StatisticUnit.Portc). |
| Filt | - | Filtro de resultados por posición, largo (StatisticFilt.Long), corto(StatisticFilt.Short) o todos (StatisticFilt.All). |
| CompType | - | Representación de los datos agrupados por negocios (StatisticCompType.Trades), días (StatisticCompType.Days), semanas (StatisticCompType.Weeks), meses (StatisticCompType.Moths) o años (StatisticCompType.Years). |
| Compressión | - | Unidad de compresión. |
| BeginDate | - | Fecha de inicio del intervalo de tiempo de las estadísticas. |
| EndDate | - | Fecha fin del intervalo de tiempo de las estadísticas. |

**Ejemplo (VB.NET):**
Supongamos que hemos definido las siguientes variables de salida:

```
Dim signo As StatisticSign
Dim unidad As StatisticUnit
Dim filtro As StatisticFilt
Dim tcomp As StatisticCompType
Dim compresion As Long
Dim fechaini As Date
Dim fechafin As Date
```

Al hacer la llamada a la propiedad GetConfigStk:

GetConfigStk(signo, unidad, filtro, tcomp, compresion, fechaini, fechafin)

Cada una de las variables definidas previamente, se rellenaran con los datos retornados por GetConfigStk.

Siguiendo con el ejemplo que se hacía para la propiedad ConfigStk, al utilizar GetConfigStk, la información que se retornaría sería:

```
signo = StatisticSign.Loss
unidad = StatisticUnit.Money
filtro = StatisticFilt.All
tcomp = StatisticCompType.Days
compresion = 1
fechaini = "01/01/2000"
fechafin = "01/01/2015"
```

## GetDailyLosers

**Descripción:**
Devuelve número de negocios perdedores entre dos fechas dadas. Se compara siempre con la fecha de entrada del negocio.

**Sintaxis:**
GetDailyLosers(FromDate, ToDate)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| FromDate | - | Fecha de inicio del intervalo a analizar. |
| ToDate | -1 | Fecha de fin del intervalo a analizar. Si no se especifica ningún valor o se asigna -1, significa que la fecha final del intervalo será la fecha actual. |

**Ejemplo (VB.NET):**

Si queremos saber el número de negocios perdedores entre dos fechas, p.e entre 10/06/2015 y 10/09/2015.

Definimos previamente la variable a la que asignaremos la cantidad de negocios perdedores y asignamos el valor que retorna la función haciendo lo siguiente:
Dim NegPerdedores As Long = Me.GetDailyLosers(“20150610”,”20150910”)

## GetDailyWinners

**Descripción:**
Devuelve el número de negocios ganadores entre dos fechas dadas. Se compara siempre con la fecha de entrada del negocio.

**Sintaxis:**
GetDailyWinners(FromDate, ToDate)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| FromDate | - | Fecha de inicio del intervalo a analizar. |
| ToDate | -1 | Fecha de fin del intervalo a analizar. Si no se especifica ningún valor o se asigna -1, significa que la fecha final del intervalo será la fecha actual. |

**Ejemplo (VB.NET):**

Si queremos saber el número de negocios ganadores entre 2 fechas, p.e entre 10/06/2015 y 10/09/2015.

Definimos previamente la variable a la que asignaremos la cantidad de negocios ganadores y asignamos el valor que retorna la función haciendo lo siguiente:

Dim NegGanadores As Long = Me.GetDailyWinners("20150610","20150910")

## GetEntryDate

**Descripción:**
Esta función se utiliza para conocer la fecha en la que se ha abierto una posición. El formato en el que se retorna la fecha es militar (**AAAAMMDD).** Por lo tanto, si la posición se abrió el día 25 de Junio de 2010, la función indicada retornará el valor numérico 20100625. Si no se ha entrado en ninguna posición, retornará el valor 0.

**Sintaxis:**
GetEntryDate(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás. El valor por defecto hace referencia a la posición actual. |

**Ejemplos (VB.NET):**

Me.GetEntryDate(0) La función devuelve la fecha en la que se abrió la posición actual.
Me.GetEntryDate(5) La función devuelve la fecha de entrada de hace 5 negocios.

## GetEntryPrice

**Descripción:**
Esta función se utiliza para conocer el precio al que ha realizado la compra/venta al abrir una posición. Retorna el precio de entrada de la posición indicada en el parámetro EntryAgo. Si no se ha entrado en ninguna posición, retorna 0.

**Sintaxis:**
GetEntryPrice(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás que deseamos consultar. El valor por defecto hace referencia a la posición actual. |

**Ejemplos (VB.NET):**

Me.GetEntryPrice(0) La función devuelve el precio de entrada de la posición actual.

Me.GetEntryPrice(5) La función devuelve el precio de entrada de hace 5 negocios.

## GetEntryTime

**Descripción:**
Esta función se utiliza para conocer la hora a la que se ha entrado en una posición. Esta información se retornará en formato militar de 24 horas **(HHMM),** de forma que si la hora es 5:35 pm se considerará como el valor numérico número 1735. Si no se ha entrado en ninguna posición, la función devuelve el valor 0.

**Sintaxis:**
GetEntryTime(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás que deseamos consultar. El valor por defecto hace referencia a la posición actual. |

**Ejemplos (VB.NET):**

Me.GetEntryTime(0) La función devuelve la hora de entrada de la posición actual.
Me.GetEntryTime(4) La función devuelve la hora de entrada de hace 4 negocios.

## GetExitDate

**Descripción:**
Esta función se utiliza para conocer la fecha en la que se ha cerrado una posición. El formato en el que se retorna la fecha es militar (**AAAAMMDD).** Si la posición se cerró el día 25 de Junio de 2010, la función indicada retornará el valor numérico 20100625. Si no se ha cerrado la posición, devolverá el valor 0.

**Sintaxis:**
GetExitDate(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|

| | | |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás. El valor por defecto hace referencia a la posición actual. |

**Observaciones:**
Hay que tener en cuenta que a todos los efectos, el último negocio abierto se considera cerrado en la barra actual (barra en la que se están realizando los cálculos). Por tanto, si se indica en el parámetro **EntryAgo** el valor de 0, esta función retornará la fecha de la barra actual.

De este modo, si en la estrategia tan sólo se utilizan las funciones Buy y Sell, el valor 1 al parámetro **EntryAgo** devolverá la fecha de la última operación cerrada antes de la actual. Mientras que el valor 0 siempre devolverá la fecha de la barra actual, indicando que actualmente existe una operación activa.

Pero si se utilizan también las funciones ExitLong y ExitShort, puede darse el caso de que en una barra no haya ningún negocio abierto. En estos casos, el valor 0 al parámetro **EntryAgo** no devolverá nada, ya que no hay ninguna operación abierta. Mientras que el valor 1 devolverá la fecha de la barra donde se liquidó la última operación.

**Ejemplo (VB.NET):**

```
Me.GetExitDate(3)        La función devuelve la fecha de salida de hace 3 negocios.
```

## GetExitOrder

**Descripción:**
Especifica si la n-ésima orden dada durante una barra concreta para una objeto de tipo estrategia es de salida o no. En el caso de haber sido una orden de salida, la función retorna **True**. Mientras que si la orden ha sido de entrada en mercado devuelve **False**.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

**Sintaxis:**
Me.Strategy.GetExitOrder(BarsAgo, NumOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la cual se desea obtener el dato. El valor por defecto hace referencia a la barra actual. |
| NumOrder | - | Posición de la orden de la que se desea obtener la información. El listado de posiciones comienza por 0. |

**Ejemplo (VB.NET):**

Queremos cerrar los negocios de la estrategia cuando la estrategia *AdxBandSystem 02* cierre.

El primer paso será declarar el objeto de tipo *ADXBANDSYS02*:

```
    Dim adxbdsysdata As ADXBANDSYS02
```

Seguidamente, creamos el objeto desde el método *OnInitCalculate*:

```
Me.adxbdsysdata = New ADXBANDSYS02(Me.Data)
```

Desde *OnCalculateBar()*, añadimos el siguiente código:

```
If (Me.adxbdsysdata.GetExitOrder(0, 0)) Then
   Me.ExitLong(TradeType.AtMarket, 1)
   Me.ExitShort(TradeType.AtMarket, 1)
End If
```

## GetExitPrice

**Descripción:**
Esta función se utiliza para conocer el precio al que se ha cerrado una posición. Si no se ha cerrado ninguna posición, retornará 0.

**Sintaxis:**
GetExitPrice(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás que deseamos consultar. El valor por defecto hace referencia a la posición actual. |

Es importante tener en cuenta las observaciones de la función GetExitDate.

**Ejemplo (VB.NET):**

```
Me.GetExitPrice(5)    La función devuelve el precio de salida de hace 5 negocios.
```

## GetExitTime

**Descripción:**

Esta función se utiliza para conocer la hora a la que se ha cerrado una posición. La hora la retornará en formato militar de 24 horas **(HHMM),** de forma que si la hora es 5:35 pm se considerará como el valor numérico número 1735. Si no se ha cerrado ninguna posición, la función devuelve el valor 0.

**Sintaxis:**
GetExitTime(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás que deseamos consultar. El valor por defecto hace referencia a la posición actual. |

**Ejemplo (VB.NET):**

Me.GetExitTime(3)  La función devuelve la hora de salida en hace 3 negocios.

Es importante tener en cuenta las observaciones de la función GetExitDate.

## GetFeedFields

**Descripción:**
Determina los campos de información que podemos obtener en tiempo real usando el método del mismo nombre.

**Sintaxis:**
GetFeedFields(Field)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Field | FFLast | **FeedFields.AskSize** Cantidad de títulos ofertados<br>**FeedFields.BidSize** Cantidad de títulos demandados<br>**FeedFields.Buy1** Precio ofertado en la primera posición de compra<br>**FeedFields.BuyAgency** Agencia compradora<br>**FeedFields.BuyOrders** Órdenes de compra<br>**FeedFields.Date** Fecha<br>**FeedFields.Decimals** Decimales<br>**FeedFields.Description** Descripción del símbolo<br>**FeedFields.Diff** Diferencia<br>**FeedFields.Diff_P** Diferencia %<br>**FeedFields.Expiry_Date** Fecha de vencimiento<br>**FeedFields.High** Máximo<br>**FeedFields.ISIN** Código ISIN<br>**FeedFields.Last** Precio último<br>**FeedFields.LastVol** Volumen último<br>**FeedFields.Low** Mínimo<br>**FeedFields.MinimumMov** Mínimo movimiento<br>**FeedFields.NumTrades** Número de negocios<br>**FeedFields.Open** Apertura<br>**FeedFields.OpenInterest** Open Interest<br>**FeedFields.Previous** Anterior<br>**FeedFields.Sell1** Precio ofertado en la primera posición de venta<br>**FeedFields.SellAgency** Agencia vendedora<br>**FeedFields.SellOrders** Órdenes de venta<br>**FeedFields.SubMarket** Submercado al que pertenece el símbolo<br>**FeedFields.Time** Hora<br>**FeedFields.Volume** Volumen |

**Ejemplo (VB.NET):**

Si queremos obtener el último negocio cerrado, previamente definimos una variable de salida y se le asignará el valor que retorne ésta propiedad:

Dim ultnegocio as Double = Me.GetFeedFields(FeedFields.Last)

## GetHighest

**Descripción:**
Esta función se utiliza para obtener el valor mayor de las últimas **n** barras (incluyendo la actual) de un objeto fuente concreto. La función está incluida en cualquier clase de tipo fuente, ya sea una serie de datos o un indicador.

**Sintaxis:**
Me.Identifier.GetHighest(Type.Price, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 1 | Número de barras hacia atrás a considerar incluyendo la barra actual. El valor por defecto 1 retornará el máximo de la barra actual. Cualquier variable de tipo numérico puede ser puesta en lugar de un número. |

**Ejemplos (VB.NET):**

Me.Data.GetHighest(Price.High, 10)

La función devuelve el valor del mayor máximo en las últimas 10 barras de la serie de datos principal (Data1).

Creamos un objeto de tipo *AvSimple* desde *OnInitCalculate.*

Me.avsimpledata = New AvSimple(Me.Data)

De modo que

Me.avsimpledata.GetHighest(Price.Close, 100)

Devuelve el valor mayor de la media móvil de las últimas 100 barras.

## GetHighestBar

**Descripción:**
Esta función se utiliza para obtener **la distancia en barras** del valor mayor de las últimas **n** barras (incluyendo

la actual) de un objeto fuente concreto. La función está incluida en cualquier clase de tipo fuente, ya sea una serie de datos o un indicador.

La distancia en barras se calcula respecto a la barra actual. De modo que si el máximo está situado en la barra actual, el valor devuelto será 0. Si está situado 15 barras hacia atrás, el valor devuelto será 15, etc…

**Sintaxis:**
Me.Identifier.GetHighestBar (Type.Price, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Tprice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 2 | Número de barras hacia atrás a considerar incluyendo la barra actual. El valor por defecto 2 retornará 0 o 1 (según donde esté el máximo). Cualquier variable de tipo numérico puede ser puesta en lugar de un número. |

**Ejemplo (VB.NET):**

Me.Data.GetHighestBar(Price.Low, 20)

La función devuelve el número de barras (hacia atrás) donde se obtiene el mayor mínimo de las 20 últimas barras de la serie de datos principal (Data1).

## GetHistogramBand

**Descripción:**
Devuelve el número de línea de banda asignado a la línea que se especifique en el parámetro Line.

**Sintaxis:**
GetHistogramBand(Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Line | - | Identifica la línea de datos de la cual se desea conocer su línea de banda. |

**Ejemplo (VB.NET):**

Supongamos que deseamos crear un indicador y a la hora de representarlo indicamos lo siguiente:

```
Me.SetIndicatorValue(x, 1)
Me.SetIndicatorValue(50, 2)
Me.SetBarRepresentation(0,1,irHistogram)
Me.SetHistogramBand(1, 2)
```

Si definimos una variable de salida y a ésta le asignamos el valor que devuelve la función:

```
Dim milineabanda as Integer = Me.GetHistogramBand(1)
```

La función retornará el valor 2 (la línea de banda).

## GetIndicatorIdentifier - GII

**Descripción:**
Esta función se utiliza para crear la serie de datos correspondiente a un indicador cualquiera y obtener un identificador de esa serie. Para ello es necesario declarar previamente una variable de tipo DataIdentifier.

Una vez definida la variable, le asignaremos el valor de la función **GetIndicatorIdentifier** para crear la serie del indicador y obtener un identificador del mismo.

El identificador del indicador debe obtenerse en el procedimiento OnInitCalculate.

Para obtener posteriormente un valor del indicador, es preciso usar la función GetIndicatorValue e indicar en el parámetro **Data** la variable en la que hemos guardado el identificador del indicador en cuestión.

El identificador obtenido por esta función puede ser utilizado además en cualquier función de VBA en la que se solicite un Data (serie de datos sobre la que se calculan las distintas funciones).

**Sintaxis:**
GetIndicatorIdentifier(Name, ParentDataIdentifier, ParamArray())

También se puede utilizar el modo abreviado **GII**:

GII(Name, ParentDataIdentifier, ParamArray())

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Name | - | Código del indicador. |
| ParentData Identifier | - | Identificador de la serie sobre la que se calculará el indicador.<br>Si indicamos en este parámetro Data, estaremos calculando el indicador sobre el data o serie sobre la que se inserte la estrategia.<br>Si deseamos obtener el identificador de un indicador que se calcula sobre otro, debemos indicar en este parámetro el identificador del indicador sobre el que deseamos que se calcule. |

| | | |
|---|---|---|
| ParamArray | - | Primer parámetro específico del indicador. Representa un grupo de parámetros cuyo número no está especificado ya que cada indicador tiene un número variable. (una media móvil tiene 2 y sin embargo el RSI tiene 3). El orden en el que deben indicarse los parámetros es el mismo en que figuran en el cuadro de diálogo que se muestra cuando vamos a insertar el indicador sobre el gráfico.<br>Si el parámetro es de tipo **Type.Price**, hace referencia a uno de los campos de la barra (Cierre, Apertura, etc.), como sería el caso de una media móvil cuyo segundo parámetro es origen de los datos.<br>En estos casos, debemos especificar el campo de la barra sobre la que se calculará el indicador. Tendremos que introducir cualquiera de las siguientes constantes que equivalen a esos campos:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen. |
| ParamArray | - | Segundo parámetro específico del indicador. |
| ... | - | ... |
| n-ésimo ParamArray | - | n-ésimo parámetro específico del indicador. |

**Ejemplo (VB.NET):**

Declaramos un objeto de tipo *DataIdentifier*:
Dim rsidata As DataIdentifier

Creamos el objeto desde *OnInitCalculate*:
Me.rsidata = Me.GetIndicatorIdentifier(Indicators.RSI, DataIdentifier.Data1, 14, 70, 30)

La fuente devuelve el identificador del indicador RSI (14,70 y 30 son los parámetros del indicador).

**Novedades Visual Chart 6:**

Visual Chart 6 pretende sacarle el máximo partido al diseño de proyectos utilizando lenguajes de programación orientado a objetos.

Como consecuencia, observaremos que podemos crear objetos de clases equivalentes al indicador que queramos incorporar.

Así, podemos definir un objeto de tipo *AvSimple*, un objeto de tipo *RSI* o un objeto de tipo *MACD*. Cada indicador creado, tanto públicos como los que hayamos creado con nuestro usuario, añade una clase nueva del mismo tipo. Gracias a esto, la inserción de indicadores en un proyecto se hace más ágil ya que no es necesario utilizar el método *GetIndicatorIdentifier*.

**Ejemplo (VB.NET):**

El objeto creado anterioremente *rsidata* también puede ser declarado de la siguiente forma:
Dim rsidata As RSI

```
Y luego lo creamos del siguiente modo:
Me.rsidata = New RSI(Me.Data)
```

## GetIndicatorPos

**Descripción:**
Esta función obtiene la tendencia que tiene un indicador en una barra determinada.

**Sintaxis:**
GetIndicatorPos( BarsAgo, Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. |
| Line | - | Identifica la línea a la que pertenece la barra de la que se quiere obtener la tendencia. |

**Ejemplo (VB.NET):**

```
If Me.GetIndicatorPos(0, 1) = IndicatorPosition.Bull Then
      MsgBox("Indicador en tendencia alcista.")
End If
```

Los valores que puede devolver esta función son los siguientes:

- **IndicatorPosition.Bull**(alcista)
- **IndicatorPosition.Bear**(bajista)
- **IndicatorPosition.Neutral**(neutral)

## GetIndicatorValue - GIV

**Descripción:**
Esta función se utiliza para obtener el valor de un indicador. Para ello es preciso previamente haber determinado el identificador del indicador en el procedimiento *OnInitCalculate* mediante la función *GetIndicatorIdentifier*.

**Sintaxis:**
GetIndicatorValue(Identifier, BarsAgo, LineNumber)

También se puede utilizar el modo abreviado **GIV**

GIV(Identifier, BarsAgo,LineNumber)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Identifier | - | Identificador del indicador. |
| BarsAgo | 0 | Representa el número de barras hacia atrás a la que deseamos hacer referencia para obtener el valor del indicador. |

| | | |
|---|---|---|
| | | Si estamos usando una media móvil, un valor igual a cero para este parámetro haría que esta función retornara el valor de la media en la barra actual.<br>Si pasamos valor 1, retornaría el valor de la media de hace una barra y así sucesivamente. |
| LineNumber | 1 | Línea del Indicador a obtener. Hay algunos indicadores que tienen más de una línea de datos. En estos casos si pasamos a este parámetro 1, devolverá un valor referente a la primera línea de datos y si pasamos 2 devolverá un valor de la segunda línea y así sucesivamente. |

**Ejemplo (VB.NET):**

Declaramos un objeto *DataIdentifier* del siguiente modo:

Dim rsidata As DataIdentifier

Creamos el objeto desde *OnInitCalculate*:

Me.rsidata = Me.GetIndicatorIdentifier(Indicators.RSI, DataIdentifier.Data1, 14, 70, 30)

La siguiente línea:

Me.GetIndicatorValue(Me.rsidata, 0, 1)

Devuelve el valor de la primera línea del indicador en la barra actual.

**Novedades Visual Chart 6:**

Visual Chart 6 pretende sacarle el máximo partido al diseño de proyectos utilizando lenguajes de programación orientado a objetos.

Como consecuencia, observaremos que podemos crear objetos de clases equivalentes al indicador que queramos incorporar.

Los objetos de tipo *Indicator* incluyen una propiedad llamada *Value* que sustituye a la función *GetIndicatorValue* explicada anteriormente.

**Sintaxis:**
Me.Indicator.Value(BarsAgo, Line)

**Ejemplo (VB.NET):**

Creamos un objeto de tipo RSI:

Me.rsidata = New RSI(Me.Data)

Desde OnCalculateBar extraemos el valor de la línea 1 del indicador en la barra actual:

Me.rsidata.Value(0, 1)

## GetLineName

**Descripción:**

Esta función devuelve el nombre que tiene la línea de datos indicada.

**Sintaxis:**

GetLineName(Line)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Line | 0 | Identifica la línea de la que se desea obtener el nombre. |

**Ejemplo (VB.NET):**

```
Me.GetLineName(1)
```

Suponiendo que estuviéramos programando un indicador y le hubiéramos asignado a la línea 2 el nombre "UpperBand", la función GetLineName(1) devolverá "UpperBand".

## GetLowest

**Descripción:**
Esta función se utiliza para obtener el valor menor de las últimas **n** barras (incluyendo la actual) de un objeto fuente concreto. La función está incluida en cualquier clase de tipo fuente, ya sea una serie de datos o un indicador.

**Sintaxis:**
Me.Identifier.GetLowest ( TPrice, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 1 | Número de barras hacia atrás a considerar incluyendo la barra actual. El valor por defecto 1 retornará el máximo de la barra actual. Cualquier variable de tipo numérico puede ser puesta en lugar de un número. |

**Ejemplo (VB.NET):**

Me.Data.GetLowest(Price.Low, 10)

La función devuelve el valor del menor mínimo en las últimas 10 barras de la serie de datos principal (Data1).

Creamos un objeto de tipo *AvSimple* desde *OnInitCalculate.*

Me.avsimpledata = New AvSimple(Me.Data)

De modo que

Me.avsimpledata.GetLowest(Price.Close, 100)

Devuelve el valor menor de la media móvil de las últimas 100 barras.

## GetLowestBar

**Descripción:**
Esta función se utiliza para obtener **la distancia en barras** del valor menor de las últimas **n** barras (incluyendo la actual) de un objeto fuente concreto. La función está incluida en cualquier clase de tipo fuente, ya sea una serie de datos o un indicador.

La distancia en barras se calcula respecto a la barra actual. De modo que si el mínimo está situado en la barra actual, el valor devuelto será 0. Si está situado 15 barras hacia atrás, el valor devuelto será 15, etc…

**Sintaxis:**
Me.Identifier.GetLowestBar (Type.Price, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Tprice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 2 | Número de barras hacia atrás a considerar incluyendo la barra actual. El valor por defecto 2 retornará 0 o 1 (según donde esté el mínimo). Cualquier variable de tipo numérico puede ser puesta en lugar de un número. |

**Ejemplo (VB.NET):**

Me.Data.GetLowestBar(Price.High, 20)

La función devuelve el número de barras (hacia atrás) donde se obtiene el menor máximo de las 20 últimas barras de la serie de datos principal (Data1).

## GetMarketPosition

**Descripción:**

Se utiliza para conocer, mientras se está calculando la estrategia, en qué posición nos encontramos, es decir, si hay abierta una posición de compra o venta, o bien estamos fuera de mercado. Esta función es muy útil si estamos trabajando con órdenes en stop o limitadas, ya que no sabemos cuándo se han ejecutando.

Los valores que puede devolver esta función son los siguientes:

- Si hay posiciones abiertas de compra (largos) devuelve 1.
- Si hay posiciones abiertas de venta (cortos) devuelve -1.
- Si no hay posiciones abiertas devuelve 0.

**Sintaxis:**

GetMarketPosition (EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de negocios hacia atrás. Por defecto es la posición actual. |

**Ejemplo (VB.NET):**

```
If (Me.GetMarketPosition(0) = 1) Then
   Me.ExitLong(TradeType.AtStop, 1, Me.GetEntryPrice() - 20)
End If
```

Si la función devuelve 1, entonces lanzamos un stop de protección para posiciones largas.

## GetMaxContracts

**Descripción:**

Esta función se utiliza para conocer el mayor número de contratos que ha habido comprados o vendidos en una posición.

**Sintaxis:**

GetMaxContracts (EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás. El valor por defecto hace referencia a la posición actual. |

**Ejemplo (VB.NET):**

```
If (Me.GetMarketPosition(0) = 1) Then
  If (Me.CurrentContracts() = Me.GetMaxContracts()) Then
    If (Me.Data.Close() > Me.GetEntryPrice() + 100) Then
      Me.ExitLong(TradeType.AtMarket, Me.CurrentContracts() / 2)
    End If
  End If
  Me.ExitLong(TradeType.AtStop, Me.CurrentContracts(), Me.GetEntryPrice() - 20)
End If
```

Si estamos abiertos a largo, buscamos realizar un objetivo parcial con la mitad de los contratos totales. Para ello, comprobamos si *CurrentContracts* es igual a *GetMaxContracts*, en caso de que ocurra, chequeamos el posible cierre parcial. Si el valor de *CurrentContracts* es inferior, entonces quiere decir que ya hemos realizado el cierre parcial y por tanto no accedemos a esta parte del código.

## GetMaxEntries

**Descripción:**
Esta función se utiliza para conocer el mayor número de entradas diferentes que ha habido en una posición. Una posición puede tener varias entradas diferentes en función de la etiqueta establecida en la orden y de la modalidad de casar operaciones elegida.

**Sintaxis:**
GetMaxEntries (EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás. El valor por defecto hace referencia a la posición actual. |

**Ejemplo (VB.NET):**

```
Me.GetMaxEntries(5)     La función devuelve el mayor número de entradas de hace 5 negocios.
```

## GetNthHighest

**Descripción:**
Esta función se utiliza para obtener el valor mayor de las últimas **n** barras con un orden concreto de una fuente de datos específica. La función está incluida en cualquier clase de tipo fuente, ya sea una serie de datos o un indicador.

**Sintaxis:**
Me.Identifier.GetNthHighest (Nth, TPrice, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Nth | 1 | Es el ordinal que representa qué valor deseamos obtener. Si Nth vale 1, obtendremos el primer valor mayor de las **n** ultimas barras de la serie, si Nth vale 2 obtendremos el segundo mayor valor de la serie y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 50 | Número de barras hacia atrás a considerar. |

**Ejemplo (VB.NET):**

```
Dim lasthigh As Double = Me.Data.GetNthHighest(1, Price.High, 100)
Dim prelasthigh As Double = Me.Data.GetNthHighest(2, Price.High, 100)
If (lasthigh > prelasthigh + 20) Then
   Me.Sell(TradeType.AtMarket, 1)
End If
```

Entramos cortos a mercado si el mayor máximo es al menos 20 puntos superior al penúltimo mayor máximo de las últimas 100 barras.

## GetNthLowest

**Descripción:**

Esta función se utiliza para obtener el valor menor de las últimas **n** barras con un orden concreto de una fuente de datos específica. La función está incluida en cualquier clase de tipo fuente, ya sea una serie de datos o un indicador.

**Sintaxis:**

Me.Identifier.GetNthLowest(Nth, TPrice, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Nth | 1 | Es el ordinal que representa qué valor deseamos obtener. Si Nth vale 1, obtendremos el primer valor menor de las **n** ultimas barras de la serie, si Nth vale 2 obtendremos el segundo menor valor de la serie y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price: |

| | | |
|---|---|---|
| | | **Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 50 | Número de barras hacia atrás a considerar. |

**Ejemplo (VB.NET):**

```
Dim lastlow As Double = Me.Data.GetNthLowest(1, Price.Low, 100)
Dim prelastlow As Double = Me.Data.GetNthLowest(2, Price.Low, 100)
If (lastlow < prelastlow - 20) Then
   Me.Buy(TradeType.AtMarket, 1)
End If
```

Entramos largos a mercado si el menor mínimo es al menos 20 puntos inferior al penúltimo menor mínimo de las últimas 100 barras.

## GetOrderCount

**Descripción:**
Devuelve la cantidad de órdenes activas dadas en una barra concreta de un objeto de tipo estrategia.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

**Sintaxis:**
Me.Strategy.GetOrderCount( BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea extraer la cantidad de órdenes. El valor por defecto hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto de tipo *DEFSYS (Default System)*:

```
Dim defsysdata As DEFSYS
```

Imaginemos que el objeto defsysdata al que queremos hacer referencia en la barra actual, envía a mercado las siguientes órdenes:

- Una orden para girarse a corto porque la estrategia está comprado

- Una orden para salirse por ganancias
- Una orden para salirse por pérdidas

Podemos asignar a una variable, previamente definida, el valor que retorna esta función:

Dim numordenes As Integer = Me.defsysdata.GetOrderCount(0)

La función retornará el valor 3 en la barra actual, ya que son las órdenes ACTIVAS en la misma.

## GetOrderDate

**Descripción:**
Devuelve la fecha asignada a la n-ésima orden activa dada una barra concreta de un objeto de tipo estrategia.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

**Sintaxis:**
Me.Strategy.GetOrderDate(Identifier, BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea extraer la fecha. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto de tipo *DEFSYS (Default System)*:

```
Dim defsysdata As DEFSYS
```

Siguiendo con el ejemplo empleado para la función GetOrderCount, si de las tres órdenes sólo nos interesa saber la fecha de la última, entonces haríamos lo siguiente:

Dim fechaulorden As Date =Me.defsysdata.GetOrderDate(0, 2)

La función retorna una fecha del tipo DD/MM/YYYY. Normalmente la fecha que devolverá será la correspondiente a la barra actual (en caso de BarsAgo=0), o a la fecha de la barra a la que hagamos referencia (en caso de BarsAgo >0).

La posición a la que hacemos referencia en este caso es 2, ya que como se indica en el apartado Parámetros de esta función, el valor 0 se asigna a la primera orden, el valor 1 a la segunda y así sucesivamente.

## GetOrderLabel

**Descripción:**
Devuelve la etiqueta asignada a la n-ésima orden activa en una barra concreta de un objeto de tipo estrategia.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

**Sintaxis:**
Me.Strategy.GetOrderLabel(BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea extraer la etiqueta. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que somos el usuario *userx* y hemos creado la estrategia *MyStrategy.* En un momento determinado, dicha estrategia realiza lo siguiente:

1) Ha entrado con una orden Buy con etiqueta "Compra_1"
2) Y prepara tres órdenes nuevas:

    a. Me.ExitLong(TradeType.Atlimit 1, GetEntryPrice + 50, "Compra_1")
    b. Me.ExitLong(TradeType.Atstop, 1, GetEntryPrice - 20, "Compra_1")
    c. Me.Sell(TradeType.Atstop, 1, .Low – 100, "Venta_1")

Partiendo de esto, tenemos una segunda estrategia en la que declaramos un objeto de la clase *userx_MyStrategy:*

Dim mystrdata As userx_MyStrategy

Dentro del método *OnCalculateBar*, indicamos lo siguiente:

Dim etiqueta As String = Me.mystrdata.GetOrderLabel(0, 1)

En etiqueta se obtendrá "Compra_1", que es la etiqueta asignada a la segunda orden de la estrategia *MyStrategy*.

## GetOrderPrice

**Descripción:**
Devuelve el precio de la n-ésima orden activa en una barra concreta de un objeto de tipo estrategia.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

**Sintaxis:**
Me.Strategy.GetOrderPrice(BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea extraer la posición. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto de tipo *DEFSYS (Default System)*:

```
Dim defsysdata As DEFSYS
```

Imaginemos que el objeto defsysdata al que queremos hacer referencia en la barra actual, envía a mercado las siguientes órdenes:

- Una orden para girarse a corto porque la estrategia está comprado
- Una orden para salirse por ganancias
- Una orden para salirse por pérdidas

Podemos asignar a una variable, previamente definida, el valor que retorna esta función:

```
Dim precio as Double= Me.defsysdata.GetOrderPrice(0, 0)
```

La función retornará el precio de la primera de las tres órdenes lanzadas en la barra actual.

## GetOrderSide

**Descripción:**
Devuelve la posición (compra o venta) de la n-ésima orden activa en una barra concreta de un objeto de tipo estrategia.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

El valor devuelto por la función será del tipo *OrderSide.* Los valores que puede tomar son:

- OrderSide.Buy
- OrderSide.Sell

**Sintaxis:**
Me.Strategy.GetOrderside(BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea obtener el símbolo del valor asociado. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto del tipo *ADXBANDSYS02*:

```
Dim adxbdsysdata As ADXBANDSYS02
```

Y ahora queremos entrar a largo cuando esta estrategia envíe una nueva orden de compra, pero a 10 puntos del precio de compra:

```
If (Me.adxbdsysdata.GetOrderSide(0, 0) = OrderSide.Buy) Then
   Me.Buy(TradeType.AtStop, 1, Me.adxbdsysdata.GetOrderPrice(0, 0) + 10)
End If
```

## GetOrderSymbolCode

**Descripción:**
Devuelve el código (en formato de Visual Chart, p.e 010072MFXI) del valor asociado de la n-ésima orden activa de un objeto de tipo estrategia, dada en una barra concreta.

Es necesario por tanto haber declarado previamente un objeto de tipo *estrategia.*

**Sintaxis:**
Me.Strategy.GetOrderSymbolCode(BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea obtener el símbolo del valor asociado. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto de tipo *DEFSYS (Default System)*:

```
Dim defsysdata As DEFSYS
```

Y a la hora de crearlo, le asignamos como fuente de referencia una serie de datos secundaria:

```
Me.defsysdata = New DEFSYS(Me.Data2)
```

Supongamos que la serie de datos secundaria fuera el Futuro del Dax Continuo. Si creamos la siguiente variable:

```
Dim simbolo as String =Me.defsysdata.GetOrderSymbolCode(0,0)
```

La función retornará “010015DX” que es el código del Futuro Dax Continuo en formato de Visual Chart.

## GetOrderType

**Descripción:**
Devuelve el tipo de orden (stop, limitada, al cierre) de la n-ésima orden activa de una estrategia, dada en una barra concreta.

**Sintaxis:**
Me.Strategy.GetOrderType(BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea extraer el tipo de orden. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto de tipo *DEFSYS (Default System)*:

```
Dim defsysdata As DEFSYS
```

Deseamos saber el tipo de una orden de esta estrategia enviada en la barra anterior. Para ello, creamos una variable de tipo Type.TradeType y le asignamos el valor devuelto por la función *GetOrderType*:

```
Dim tipo As TradeType = Me.defsysdata.GetOrderType(1, 0)
```

La variable almacenará un valor de tipo *Type.AtMarket* .

## GetOrderVolume

**Descripción:**
Devuelve la cantidad de contratos/acciones de la n-ésima orden activa de una estrategia, dada en una barra concreta.

**Sintaxis:**
Me.Strategy.GetOrderVolume( BarsAgo, NumberOrder)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|

| | | |
|---|---|---|
| BarsAgo | 0 | Barra de la que se desea extraer el volumen. El valor por defecto hace referencia a la barra actual. |
| Number Order | 0 | Número de orden de referencia asignado a la orden de la que se quiere extraer la información. El valor por defecto hace referencia a la primera orden activa que se ha realizado una barra concreta. |

**Ejemplo (VB.NET):**

Supongamos que declaramos un objeto de tipo *DEFSYS (Default System)*:

```
Dim defsysdata As DEFSYS
```

Deseamos saber el volumen de contratos/acciones de la segunda orden en la barra actual. Para ello, definimos una variable y asignamos a esta el valor que retorne la función.

```
Dim volorden As Long = Me.defsysdata.GetOrderVolume(0,0)
```

volorden almacenará el número de contratos asociados a la orden actual.

## GetPivotDown

**Descripción:**
Esta función se utiliza para obtener el valor de un pivote. Un pivote (o swing) es un mínimo que se dió en la cotización y que representó un giro del mercado al alza.

**Sintaxis:**
Me.Identifier.GetPivotDown(Occur, TPrice, LeftCount, RightCount, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Occur | 1 | Valor numérico que representa el número de pivote hacia atrás que deseamos obtener. Si Occur vale 1, estaremos obteniendo el primer pivote que haya desde la barra actual, si vale 2 el segundo pivote y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| LeftCount | - | Es el número de barras a la izquierda del pivot. |
| RightCount | - | Es el número de barras a la derecha del pivot. |

| | | |
|---|---|---|
| Length | 50 | Número de barras hacia atrás a considerar en la búsqueda del pivot. |

**Ejemplo (VB.NET):**

Me.Data.GetPivotDown(1, Price.Low, 2,4, 50)

Buscará, en las 50 barras anteriores a la actual, el valor del pivot más cercano (calculado sobre los mínimos), siendo en las 2 barras a la izquierda del pivot, y en las 4 barras a la derecha de éste, el valor del mínimo superior al de éste.

## GetPivotUp

**Descripción:**
Esta función se utiliza para obtener el valor de un pivot. Un pivote (o swing) es un máximo que se dió en la cotización y que representó un giro del mercado a la baja.

**Sintaxis:**
Me.Identifier.GetPivotUp(Occur, TPrice, LeftCount, RightCount, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Occur | 1 | Valor numérico que representa el número de pivote hacia atrás que deseamos obtener. Si Occur vale 1, estaremos obteniendo el primer pivote que haya desde la barra actual, si vale 2 el segundo pivote y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| LeftCount | - | Es el número de barras a la izquierda del pivot. |
| RightCount | - | Es el número de barras a la derecha del pivot. |
| Length | 50 | Número de barras hacia atrás a considerar en la búsqueda del pivot. |

**Ejemplo (VB.NET):**

Me.Data.GetPivotUp( 1 , Price.High, 2,4, 50)

Buscará, en las 50 barras anteriores a la actual, el valor del pivot más cercano (calculado sobre los máximos)

siendo en las 2 barras a la izquierda del pivot, y en las 4 barras a la derecha de éste, el valor del máximo inferior al de éste.

## GetPositionProfit

**Descripción:**
Se utiliza para conocer el valor (monetario) de la ganancia que se ha obtenido en una posición. Esta función considera el número de contratos/acciones que hayamos comprado/ vendido.

El valor que retorne será la diferencia entre el precio de salida y el precio de entrada (teniendo en cuenta el signo de la operación), multiplicado por el número de contratos/acciones. A dicho resultado se le resta la penalización total aplicada a la estrategia.

Si este valor es positivo, la función indicará beneficio, y en caso de ser negativo, pérdida.

Si el parámetro *EntryAgo* vale 0, se aplica la función al negocio actual. De modo que:

- Si en dicho momento no hay negocios abiertos, entonces el valor devuelto será 0.
- Si hay un negocio abierto, devolverá la ganancia obtenida hasta el momento actual, esto es, tomando como punto de salida el cierre de la última barra.

Si el parámetro *EntryAgo* vale 1, la función retornará la ganancia del último negocio finalizado.

**Sintaxis:**
GetPositionProfit(EntryAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| EntryAgo | 0 | Número de posiciones hacia atrás. El valor por defecto indica la posición actual. |

**Ejemplo (VB.NET):**

Queremos tomar como objetivo para el negocio actual la pérdida del último negocio, en caso de haberla.

```
If (Me.GetMarketPosition() = 1) Then
   Dim ganptos As Double = (Me.GetPositionProfit(1) / Me.Data.GetSymbolInfo(SymbolInfo.PointValue))
   If (ganptos < 0) Then
      Me.ExitLong(TradeType.AtLimit, 1, Me.GetEntryPrice() + Math.Abs(ganptos))
   End If
End If
```

La ganancia se divide por el valor por punto para obtener su valor en puntos en lugar del valor monetario.

## GetPrice

**Descripción:**
Esta función se utiliza para conocer el precio de un campo de una barra determinada perteneciente a una serie de datos.

**Sintaxis:**
Me.Identifier.GetPrice(TPrice, BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen. |
| BarsAgo | 0 | Número de barras hacia atrás. |

**Ejemplo (VB.NET):**

Me.Data.GetPrice(PriceHigh, 22) Devuelve el precio máximo de 22 barras atrás, de la serie Data1.

## GetStkLength

**Descripción:**
Devuelve la cantidad total de valores dados para un tipo de dato estadístico concreto.

**Sintaxis:**
GetStkLength (statistic)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| statistic | - | Permite seleccionar el estadístico del que se quiere extraer el valor. Este parámetro es del tipo *Types.StatisticValue* |

**Ejemplo (VB.NET):**

Dentro de una estrategia, queremos saber en un momento determinado, la longitud del Drawdown, es decir, cuantos datos (negocios) lleva calculados. Haríamos lo siguiente:

Dim longitud_dd As Double = Me.GetStkLength(StatisticValue.Drawdown)

Longitud_dd devolverá la cantidad de negocios que lleva la estrategia en el momento de llamar a la función, y que se han usado para calcular el drawdown.

## GetStkValue

**Descripción:**

Devuelve el valor n-ésimo de un tipo de dato estadístico concreto. El valor devuelto puede estar definido en puntos, en valor monetario o en términos porcentuales (definición por defecto).

**Sintaxis:**
GetStkValue(statistic, index)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| statistic | - | Permite seleccionar el estadístico del que se quiere extraer el valor. Este parámetro es del tipo *Types.StatisticValue* |
| index | - | Posición n-ésima desde la que se quiere obtener el estadístico. Cada posición hace referencia a un negocio de la estrategia. |

**Acerca del parámetro *index*:**

Considerando la posición 0 como el valor del estadístico en el momento actual, cada incremento de posición supondrá una mayor distancia. Así, el valor del estadístico en la posición 1 hace referencia al dato del estadístico en el momento de cierre del penúltimo negocio, mientras que la posición 2 hará referencia al dato del estadístico en el momento de cierre del antepenúltimo negocio, y así constantemente.

**Ejemplo (VB.NET):**

La ganancia porcentual neta de una estrategia en un momento determinado es 0.823%, de modo que si declaramos la siguiente variable:

Dim neto_actual As Double = Me.GetStkValue(StatisticValue.NetProfit, 0)

El valor de neto_actual será 0.823.

El último negocio cerrado de dicha estrategia obtuvo una resultado de -225€, que en términos porcentuales supone un -0.088% de pérdida, de modo que si declaramos la siguiente variable:

Dim neto_previo As Double = Me.GetStkValue(StatisticValue.NetProfit, 1)

El valor de neto_previo será 0.911. Es decir, la ganancia porcentual neta antes de cerrar el último negocio.

## GetStkValues

**Descripción:**
Devuelve el conjunto de valores totales para un tipo de dato estadístico concreto.

**Sintaxis:**
GetStkValues(statistic, aValues)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| statistic | - | Permite seleccionar el estadístico del que se quiere extraer el valor. Este parámetro es del tipo *Types.StatisticValue* |
| aValues | - | Búffer donde se almacenan los valores retornados para el tipo de dato estadístico solicitado. |

**Ejemplo (VB.NET):**

```
Dim cantidad As Long = 0
Dim values() As Double = Nothing

cantidad = Me.GetStkValues(StatisticValue.Drawdown, values)
```

En este caso, devuelve en el array **values** todos los datos. Y en **cantidad** la cantidad de valores escritos en el array.

## GetSymbolIdentifier - GSI

**Descripción:**
Esta función es necesaria sólo en los casos en los que deseemos utilizar los datos de un símbolo que no esté en pantalla a la hora de insertar la estrategia de que se trate, pues si así fuera, sería más cómodo utilizar los **Datas** correspondientes.

Esta función también es útil para hacer referencia a valores de un símbolo determinado en una Macro sobre Tabla, pues en este caso no tenemos disponibles los datos históricos en pantalla.

Para crear una fuente de datos y obtener el identificador de la misma es preciso declarar previamente una variable de tipo *DataIdentifier*.

Una vez definida la variable, le asignaremos el valor de la función **GetSymbolIdentifier** para obtener el identificador del símbolo. El identificador del símbolo debe obtenerse en el procedimiento *OnInitCalculate*.

El identificador obtenido por esta función puede ser utilizado luego en cualquier función o propiedad en la que se nos solicite un Data (serie de datos sobre la que se calculan las distintas funciones).

**Sintaxis:**
GetSymbolIdentifier(Symbol, Compresion, Cr, FromDate, ToFinalDate)
También se puede utilizar su modo abreviado **GSI:**
GSI(Symbol, Compresion, Cr, FromDate, ToFinalDate)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Symbol | - | Código del símbolo deseado. |
| Compresion | - | Unidad de compresión (2, 5 ,10...). |
| Cr | - | Tipo de compresión. E valor solicitado es del tipo *Types.CompresionRange*:<br>**CompresionRange.Minutos:** Para obtener un gráfico de minutos.<br>**CompresionRange.Dias:** Para obtener un gráfico diario.<br>**CompresionRange.Semanas:** Para obtener un gráfico de barra semanal.<br>**CompresionRange.Meses:** Para obtener un gráfico de barra mensual. |
| FromDate | - | Fecha inicial de la fuente de datos cuyo identificador vamos solicitamos. |
| ToFinalDate | - | Es la fecha final de los datos históricos que vamos a cargar. Normalmente esta fecha deseamos que sea siempre superior a la fecha actual por lo que |

| | | |
|---|---|---|
| | | aconsejamos que se indique en este parámetro la siguiente fecha "01/01/2037" para asegurarnos de que siempre se actualizan los datos de la fuente cuyo identificador solicitamos. |

**Ejemplo (VB.NET):**

```
Dim dailydate As DataIdentifier = Me.GetSymbolIdentifier(Me.Data.GetSymbolInfo(SymbolInfo.Code), 1, CompresionRange.Dias, "01/01/1997", "01/01/2037")
```

Declaramos un objeto de tipo *DataIdentifier* y al que le asignamos la serie de datos diaria del mismo símbolo que la fuente principal.

## GetSymbolInfo

**Descripción:**
Esta función devuelve una serie de características propias del producto en cuestión, y no sólo de una barra en concreto. Estos valores permanecen constantes durante todo el gráfico. La información que retorna la función viene determinada por el tipo de valor **Types.SymbolInfo**, el cual puede tomar los siguientes valores:

| | |
|---|---|
| SymbolInfo.BarCompresion | Compresión utilizada para las barras. |
| SymbolInfo.Code | Código del activo. |
| SymbolInfo.Compresion | Tipo de compresión que se está usando (ticks, minutos, días,…) |
| SymbolInfo.FirstSessionEnd | Hora de cierre de sesión del activo (formato HHMM). |
| SymbolInfo.FirstSession Start | Hora de inicio de sesión del activo (formato HHMM). |
| SymbolInfo.Market | Mercado al que pertenece el producto. |
| SymbolInfo.MinMov | Mínimo movimiento (pipos) que puede llegar a dar el producto. |
| SymbolInfo.Name | Nombre del producto. |
| SymbolInfo.NumDec | Número de decimales considerados en la escala que usa el producto. |
| SymbolInfo.Path | Todos los símbolos registrados en nuestro ordenador se encuentran en la carpeta RealServer\Data de VisualChart. Con este dato se puede extraer la ruta dentro de la carpeta Data de un activo en cuestión. |
| SymbolInfo.PointValue | Valor por punto del producto. |
| SymbolInfo.TimeD if | Diferencia horaria del producto sobre el que se aplica (en segundos) |
| SymbolInfo.Vendor | Especifica el Vendor del activo. |

**Sintaxis:**

```
Me.Identifier.GetSymbolInfo(SymbolInfo)
```

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Info | SymbolInfo.Name | Tipo de SymbolInfo que deseamos consultar. |

**Ejemplo (VB.NET):**

Declaramos una variable global de tipo lógico que nos dirá si actuamos en una compresión temporal *intradiaria.*

```
Dim esintradia As Boolean
```

Desde el método *OnInitCalculate*, le damos valor a la variable en base al tipo de información SymbolInfo.Compresion:

```
If Me.Data.GetSymbolInfo(SymbolInfo.Compresion) = CompresionRange.Minutos Or
Me.Data.GetSymbolInfo(SymbolInfo.Compresion) = CompresionRange.Ticks Then
  esintradia = True
Else
  esintradia = False
End If
```

Es decir, que si la compresión es de minutos o tics, estamos en un gráfico intradiario, en otro caso, no.

## GetSymbolInfoEx

**Descripción:**
Esta propiedad devuelve un objeto de la clase **IElementSymbolInfo** relativa a la serie de datos asignada en el momento de crear la clase. El uso de esta propiedad es el de evitar tener que hacer más de una llamada al recurso para extraer todos los datos **SymbolInfo**.

Una vez creado el objeto, se puede hacer uso de sus funciones para poder acceder a los datos de tipo *SymbolInfo*. Las funciones que podemos usar de dicha clase son las siguientes:

| | |
|---|---|
| GetBarCompresion | Devuelve la compresión utilizada para las barras. |
| GetCode | Devuelve el código del activo. |
| GetCompresion | Devuelve el tipo de compresión que se está usando (ticks, minutos, días,…) |
| GetFirstSessionEnd | Devuelve la hora de cierre de sesión del activo (formato HHMM). |
| GetFirstSession Start | Devuelve la hora de inicio de sesión del activo (formato HHMM). |
| GetMarket | Devuelve el mercado al que pertenece el producto. |
| GetMinMov | Devuelve el mínimo movimiento (pipos) que puede llegar a dar el producto. |
| GetName | Devuelve el nombre del producto. |
| GetNumDec | Devuelve el número de decimales considerados en la escala que usa el producto. |
| GetPath | Devuelve todos los símbolos registrados en nuestro ordenador se encuentran en la carpeta RealServer\Data de VisualChart. Con este dato se puede extraer la ruta dentro de la carpeta Data de un activo en cuestión. |
| GetPointValue | Devuelve el valor por punto del producto. |
| GetTimeD if | Devuelve la diferencia horaria del producto sobre el que se aplica (en segundos) |
| GetVendor | Devuelve el vendor del activo. |

**Sintaxis:**

```
Dim x As IElementSymbolInfo = Me.GetSymbolInfoEx(DataIdentifier, [Day])
```

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Symbol | Data | Fuente de datos de la que se quiere extraer la información. Si no se especifica, se considera la fuente de datos sobre el que se inserta la estrategia (con el alias de Data1). |
| Day | - | Parámetro opcional. Día de la semana del que se quiere extraer la información. Normalmente no se declara. |

**Ejemplo (VB.NET):**

Creamos un objeto de la clase IElementSymbolInfo:

```
Dim ielements As IElementSymbolInfo = Me.GetSymbolInfoEx(Me.Data2.DataSeriesId)
```

La fuente de datos de referencia es la fuente secundaria asociada al alias Data2. Como lo que se solicita es un *DataIdentifier* en el parámetro *Symbol*, especificamos que la referencia es la propiedad *DataSeriesId* de Data2.

Seguidamente, indicamos que si la fuente con el alias Data2 es el futuro del Dax continuo, entonces que envíe un mensaje de advertencia:

```
If ielements.GetCode = "010015DX" Then
    MsgBox("Está usando el dax futuro continuo como Data2.")
End If
```

## GetSystemIdentifier - GSYSI

**Descripción:**
Es una función que permite obtener  internamente la información de una estrategia determinada. De este modo, se puede extraer información de dicha estrategia sin necesidad de calcularla nuevamente.

**Sintaxis:**
GetSystemIdentifier(Name, ParentDataIdentifier, ParamArray)

También se puede utilizar el método abreviado **GSYSI.**

GSYSI(Name, ParentDataIdentifier, ParamArray)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Name | - | Código de la estrategia del cual se desea obtener la información. |
| ParentData Identifier | Data | Fuente de datos sobre la que se estaría cargando la estrategia. |
| ParamArray | - | Colección de parámetros de entrada que exige la estrategia (opcional). |

**Ejemplo (VB.NET):**

Podemos asignar a una variable de tipo DataIdentifier la información de una estrategia determinada, y utilizarla para el uso de otras funciones, por ejemplo, GetOrderCount,GetOrderLabel, etc…

Supongamos que tenemos una estrategia que se llama *MiSistema*.

```
Dim mysystemdata As DataIdentifier = 0

mysystemdata =GetSystemIdentifier(MiSistema, Data, NumeroContratos, Objetivo, Perdidas, HoraInicio, HoraFin)
```

Desde *OnCalculateBar* extraemos el número de órdenes de la estrategia en un momento determinado:

Dim numorder As Long =Me.GetOrderCount(Me.mysystemdata)

**Novedades Visual Chart 6:**

Visual Chart 6 pretende sacarle el máximo partido al diseño de proyectos utilizando lenguajes de programación orientado a objetos.

Como consecuencia, observaremos que podemos crear objetos de clases asociadas a las estrategias tanto públicas como privadas (siempre que tengamos acceso a ellas).

Así, podemos definir un objeto de tipo *ADXBANDSYS*, un objeto de tipo *STOCHCROSS* o un objeto de tipo *PVBREAK*. Cada una de las estrategias añade una clase nueva del mismo tipo. Gracias a esto, la inserción de estrategias en un proyecto se hace más ágil ya que no es necesario utilizar el método *GeSystemIdentifier*.

**Utilización de la línea de ganancia.**
Los objetos de clases tipo *estrategia* también son considerados fuentes de datos, sólo que en estos casos la serie la conforma la curva de ganancia en puntos de la estrategia en concreto. Esta capacidad permite incorporar a un proyecto cualquier indicador calculado sobre la curva de ganancia de una estrategia.

**Ejemplo (VB.NET):**

Queremos aplicar el RSI de 20 periodos a la curva de ganancia de la estrategia Stochastic Cross y actuar en función de dicho RSI. Para ello, declaramos un objeto de tipo *STOCHCROSS* y otro de tipo *RSI.*

*Dim stoccrdata As STOCHCROSS*

*Dim rsidata As RSI*

Desde *OnInitCalculate*, creamos ambos objetos:

```
Me.stoccrdata = New STOCHCROSS(Me.Data, 14, 6, 3, 74, 23, 0.2, 0.6, 0, 0, 1)
Me.rsidata = New RSI(Me.stoccrdata, 20)
```

Y por ultimo desde *OnCalculateBar* actuamos en base al RSI:

```
If (Me.rsidata.value() > 30 And Me.rsidata.value(1) <= 30) then
    Me.Buy(TradeType.AtMarket)
End If
```

## GetSwingHigh

**Descripción:**
Esta función se utiliza para obtener el valor de un pivot. Un pivote (o swing) es un máximo que se dió en la cotización y que representó un giro del mercado a la baja.

La diferencia respecto a *GetPivotUp* es que en esta ocasión el número de barras a derecha e izquierda es el mismo y están definidos por el parámetro *Strength*.

La función devuelve el valor nulo si no localiza un pivote alcista dentro de los requisitos específicos.

**Sintaxis:**
Me.Identifier.GetSwinHigh(Occur, Tprice, Strength, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Occur | 1 | Valor numérico que representa el número de pivote hacia atrás que deseamos obtener. Si Occur vale 1, estaremos obteniendo el primer pivot que haya desde la barra actual, si vale 2 el segundo pivote y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Strength | 2 | Número de barras a considerar ambos lados del pivote. |
| Length | 50 | Número de barras hacia atrás a considerar en la búsqueda del pivote. |

Esta función es de gran utilidad pues nos ayuda a buscar puntos de resistencia o figuras de vuelta. Decimos que existe un pívot Superior cuando un valor de una serie de datos es mayor que un número de valores anteriores y posteriores especificados en el parámetro Strength (fuerza del pivote).

**Ejemplo (VB.NET):**

Me.Data.GetSwingHigh(1, Price.High, 2, 50)

Buscará, en las 50 barras anteriores a la actual, el valor del pivot más cercano (calculado sobre los máximos), siendo en las 2 barras a cada lado del pivot, el valor del máximo inferior al de éste.

## GetSwingHighBar

**Descripción:**
Esta función se utiliza para obtener la distancia hasta un pivote alcista. Un pivote (o swing) es un máximo que se dió en la cotización y que representó un giro del mercado a la baja.

La función devuelve el valor nulo si no localiza un pivote alcista dentro de los requisitos específicos.

**Sintaxis:**
Me.Identifier.GetSwinHighBar(Occur, Tprice, Strength, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Occur | 1 | Valor numérico que representa el número de pivote hacia atrás que deseamos obtener. Si Occur vale 1, estaremos obteniendo el primer pivote que haya desde la barra actual, si vale 2 el segundo pivote y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Strength | 5 | Número de barras a considerar ambos lados del pívot para que se considere como tal. |
| Length | 50 | Número de barras hacia atrás a considerar en la búsqueda del pívot. Si se supera el número de barras la función devuelve el valor Nulo. |

**Ejemplo (VB.NET):**

```
Me.Data.GetSwingHighBar(1, Price.High, 2, 50)
```

Buscará, en las 50 barras anteriores a la actual, la distancia hasta el pivot más cercano (calculado sobre los máximos), siendo en las 2 barras a cada lado del pivot, el valor del máximo inferior al de éste. Si dicho pivot se encuentra a 5 barras desde la barra actual en la que se usa la función, devolverá un 5. En la barra siguiente, devolverá un 6 y así sucesivamente hasta que aparezca un nuevo pivot o bien la distancia supere las 50 barras máximas.

## GetSwingLow

**Descripción:**

Esta función se utiliza para obtener el valor de un pivote bajista. Un pivote (o swing) es un mínimo que se dió en la cotización y que representó un giro del mercado al alza.

La diferencia respecto a *GetPivotDown* es que en esta ocasión el número de barras a derecha e izquierda es el mismo y están definidos por el parámetro *Strength*.

La función devuelve el valor nulo si no localiza un pivote bajista dentro de los requisitos específicos.

**Sintaxis:**

Me.Identifier.GetSwingLow( Occur, Tprice, Strength, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|

| | | |
|---|---|---|
| Occur | 1 | Valor numérico que representa el número de pivote hacia atrás que deseamos obtener. Si Occur vale 1, estaremos obteniendo el primer pivote que haya desde la barra actual, si vale 2 el segundo pivote y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Strength | 2 | Número de barras a considerar ambos lados del pivote. |
| Length | 50 | Número de barras hacia atrás a considerar en la búsqueda del pivote. |

Esta función es de gran utilidad pues nos ayuda a buscar soportes o figuras de vuelta. Decimos que existe un pivote inferior cuando un valor de una serie de datos es menor o igual a un número de valores anteriores y posteriores especificados en el parámetro Strength.

**Ejemplo (VB.NET):**

```
Me.Data.GetSwingLow( 1, Price.Low, 2, 50)
```

Buscará, en las 50 barras anteriores a la actual, el valor del pivot más cercano (calculado sobre los mínimos), siendo en las 2 barras a cada lado del pivot, el valor del mínimo superior al de éste.

## GetSwingLowBar

**Descripción:**
Esta función se utiliza para obtener la distancia hasta un pivote bajista. Un pivote (o swing) es un mínimo que se dió en la cotización y que representó un giro del mercado al alza.

La función devuelve el valor nulo si no localiza un pivote bajisa dentro de los requisitos específicos.

**Sintaxis:**
Me.Identifier.GetSwingLowBar(Occur, Tprice, Strength, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Occur | 1 | Valor numérico que representa el número de pivote hacia atrás que deseamos obtener. Si Occur vale 1, estaremos obteniendo el primer pivote que haya desde la barra actual, si vale 2 el segundo pivote y así sucesivamente. |
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price: |

| | | |
|---|---|---|
| | | **Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Strength | 2 | Número de barras a considerar ambos lados del pívot para que se considere como tal. |
| Length | 50 | Número de barras hacia atrás a considerar en la búsqueda del pívot. Si se supera el número de barras la función devuelve el valor Nulo. |

**Ejemplo (VB.NET):**

```
Me.Data.GetSwingLow( 1, Price.Low, 2, 50)
```

Buscará, en las 50 barras anteriores a la actual, la distancia hasta el pivot más cercano (calculado sobre los mínimos), siendo en las 2 barras a cada lado del pivot, el valor del mínimo superior al de éste. Si el pivot se encuentra a 5 barras desde la barra actual en la que se usa la función, devolverá un 5. En la barra siguiente, devolverá un 6 y así sucesivamente hasta que aparezca un nuevo pívot.

## GetTrueHigh

**Descripción:**
Devuelve el precio más alto entre la máxima de una barra y el cierre de la barra anterior.

**Sintaxis:**
Me.Identifier.GetTrueHigh (BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 10 | Número de barra hacia atrás que servirá de referencia para el cálculo. |

**Ejemplo (VB.NET):**

Queremos tomar como precio de referencia para entrar largos en stop el valor más alto entre la máxima de la penúltima barra y el cierre de la antepenúltima barra:

```
Me.Buy(TradeType.AtStop, 1, Me.Data.GetTrueHigh (1) )
```

## GetTrueLow

**Descripción:**
Devuelve el precio más bajo entre la mínima de una barra y el cierre de la barra anterior.

**Sintaxis:**
Me.Identifier.GetTrueLow (BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 10 | Número de barras hacia atrás que servirá como referencia para el cálculo. |

**Ejemplo (VB.NET):**

Me.Data.GetTrueLow ( 5)

Compara en el histórico de la serie principal, el mínimo de una barra con el cierre de la anterior, indicándose el mínimo de los 2 (tomando como referencia para comenzar el cálculo 5 barras hacia atrás).

## GetTrueRange

**Descripción:**
Retorna la diferencia entre las funciones GetTrueHigh y GetTrueLow en una serie de datos.

**Sintaxis:**
Me.Identifier.GetTrueRange (BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 10 | Número de barra sobre la que se realiza el cálculo. |

**Ejemplo (VB.NET):**

Me.Data.GetTrueRange ( 2)

Retorna la diferencia entre el valor que devuelve la función GetTrueHigh y GetTrueLow en la segunda barra hacia atrás, de la serie Data.

## GetTrueRangeCustom

**Descripción:**
Permite comparar los niveles máximo y mínimo actuales con un intervalo de precios establecido mediante parámetros. Si el precio se mueve dentro de dicho intervalo, el valor devuelto será el rango o diferencia entre el valor máximo y el valor mínimo. Si el precio supera el valor máximo, el valor devuelto será el rango o diferencia entre el máximo dado por el activo y el valor mínimo del intervalo. Igualmente, si lo que se supera es el valor mínimo, el valor devuelto será el rango o diferencia entre el máximo del intervalo y el precio mínimo dado por el activo. De esta manera, se sabe rápidamente si el subyacente se encuentra fuera del intervalo y en qué proporción si el valor devuelto por la función supera el rango por defecto.

**Sintaxis:**
Me.DataIdentifier.GetTrueRangeCustom (BarsAgo, High, Low)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 10 | Número de barra de referencia a comparar del subyacente. |
| High | - | Valor máximo prestablecido del intervalo. |
| Low | - | Valor mínimo prestablecido del intervalo. |

**Ejemplo (VB.NET):**

Supongamos que creamos un indicador que hace lo siguiente:

```
Dim diferencia As Double = Me.Data.GetTrueRangeCustom(0, 9000, 5000)

Me.SetIndicatorValue(diferencia)
```

Insertamos el indicador sobre el gráfico del DAX FUT. CONTINUO.

Entre 2010 y 2014, el valor devuelto por el indicador será de 4000 puntos, mientras que en 2014 el valor devuelto supera los 4000 puntos y a partir de 2015 lo supera ampliamente (hasta alcanzar 7000 de rango, que es la diferencia entre los máximos hechos por el DAX en 2015 y el valor mínimo 5000).

## GetVolatility

**Descripción:**
Devuelve la volatilidad entre la barra actual y la n-ésima barra hacia atrás, en términos de diferencia de puntos.

**Sintaxis:**
Me.Identifier.GetVolatility (Tprice, Length)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| TPrice | Price.Close | Campo de la barra al que se desea hacer referencia. Para ello se debe indicar en este campo cualquier enumerador de la enumeración Types.Price:<br>**Price.High:** Equivale a la Máxima.<br>**Price.Low:** Equivale a la Mínima.<br>**Price.Open:** Equivale a la Apertura.<br>**Price.Close:** Equivale al Cierre.<br>**Price.Volume:** Equivale al Volumen.<br>Si se calcula esta función sobre un indicador, se pasará como parámetro en **TPrice** el valor **Price.Close**, que hace referencia al cierre de la serie de datos del identificador. Si se indicara cualquier otro sería indiferente, pues retornaría siempre el mismo valor. |
| Length | 1 | Distancia con respecto a la barra actual para calcular la volatilidad. |

**Ejemplo (VB.NET):**

Me.GetVolatility(Price.Low, 5)

Devuelve la diferencia entre mínimo de la barra actual y el de hace 5 barras (de la serie de datos Data1).

## GetWndBackGroundColor

**Descripción:**
Retorna el color de fondo de la ventana de un indicador.

**Sintaxis:**
GetWndBackGrounColor()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

## GrossLoss

**Descripción:**
Obtiene el valor de las pérdidas brutas de los negocios negativos que nuestro sistema lleva acumulado hasta la barra actual (en la que se están realizando los cálculos). Para obtenerla consideramos que el último negocio abierto concluye en el cierre de la barra actual.

**Sintaxis:**
GrossLoss(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Me.GrossLoss(SttRepresentation.Porcentual)

Retornará el porcentaje de pérdida (bruta) obtenida por la estrategia.

## GrossProfit

**Descripción:**
Obtiene el valor de la ganancia brutas de los negocios positivos que nuestro sistema lleva acumulado hasta la barra actual (barra en la que se están realizando los cálculos). Para obtenerla consideramos que el último negocio abierto concluye en el cierre de la barra actual.

**Sintaxis:**
GrossProfit(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Me.GrossProfit(SttRepresentation.Puntos)

Retornará en puntos la ganancia bruta obtenida por la estrategia.

## High

**Descripción:**
Esta función devuelve el valor de la máxima de una barra de una serie de datos concreta.

**Sintaxis:**
Me.Identifier.High(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hacer referencia a la barra actual.<br>En este parámetro se puede utilizar cualquier valor numérico contenido en una variable o introducir directamente el valor.<br>Puede especificarse como función sustituyendo el valor numérico. |

**Ejemplo (VB.NET):**

Me.Data.High(4) Retornará el máximo de 4 barras atrás (de la fuente de datos Data1).

## IsFirstDayBar

**Descripción:**

Esta propiedad devuelve verdadero o falso en base a si la barra actual es o no (respectivamente) la primera barra de cada sesión.

**Sintaxis:**
Me.IsFirstDayBar()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | |

**Ejemplo (VB.NET):**

Queremos entrar en base al precio de apertura de cada sesión +/- un filtro. Para ello, creamos una variable donde almacenar el precio de apertura de la sesión:

```
Private entryprice As Double
```

Desde el método *OnCalculateBar()*, indicamos que actualice la variable cada vez que la propiedad *IsFirstDayBar* sea cierta:

```
If (Me.IsFirstDayBar) Then
   Me.entryprice = Me.Data.Open()
End If
```

## IsLastDayBar

**Descripción:**

Esta propiedad devuelve verdadero o falso en base a si la barra actual es o no (respectivamente) la última barra de cada sesión.

**Sintaxis:**

Me.IsLastDayBar()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | |

**Ejemplo (VB.NET):**

Queremos que la estrategia compruebe las reglas de entrada en todas las barras a excepción de la última de cada sesión, ya que deseamos evitar que quede una orden pendiente de un día para otro:

```
If (Me.IsLastDayBar = False) Then
   If (Me.rsidata.Value() < 70 And Me.rsidata.Value(1) >= 70) Then
      Me.Buy(TradeType.AtMarket)
   End If
End If
```

## LargestLosingTrade

**Descripción:**

Devuelve el resultado de la peor operación realizada. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
LargestLosingTrade(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Deseamos saber en un momento determinado, el resultado de la peor operación (en puntos) realizada por nuestro sistema. Podríamos definir una variable:

```
Dim peoroper As Double = 0
```

Y a esta asignarle el valor que devuelve la propiedad:

```
peoroper =Me.LargestLosingTrade(SttRepresentation.ByPoints)
```

## LargestWinningTrade

**Descripción:**
Devuelve el resultado de la mejor operación realizada. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
LargestWinningTrade(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Deseamos saber en un momento determinado, el resultado de la mejor operación (en porcentaje) realizada por nuestro sistema. Podríamos definir una variable:

```
Dim mejoroper As Double = 0
```

Y a esta asignarle el valor que devuelve la propiedad:

```
mejoroper  = Me.LargestWinningsTrade(SttRepresentation.Porcentual)
```

## Lc_Index

**Descripción:**
Devuelve el ratio **Ganancias a Largo/Ganancias a Corto**. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
Lc_Index(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Deseamos saber en un momento determinado, el ratio ganancias a largo/ganancias a corto (porcentual) de nuestro sistema. Podríamos definir una variable:

```
Dim ratio_ganancia As Double = 0
```

Y a esta asignarle el valor que devuelve la propiedad:

```
ratio_ganancia=Me.Lc_Index(SttRepresentation.Porcentual)
```

## LimitOrder

**Descripción:**
Esta propiedad permite obtener la cantidad existente de órdenes activas de oferta y demanda para una posición concreta, en una barra determinada.

Precaución. Sólo devuelve resultados durante el tiempo real, pues no existe histórico de las posiciones de oferta y demanda.

**Sintaxis:**
LimitOrder(Level, Side, BarsAgo, Identifier)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Level | 1 | Indica la n-ésima posición de oferta/demanda de la que deseamos saber la cantidad. |
| Side | OrderSide.Buy | Posición de la orden . Dato de tipo Type.OrderSide, pudiendo ser:<br>**OrderSide.Buy** (demanda)<br>**OrderSide.Sell** (oferta) |
| BarsAgo | 0 | Posición de la barra que se quiere consultar, aunque debe ser una barra generada durante el tiempo real. |
| Identifier | Data | Serie de datos de la que se extrae la información. |

**Ejemplo (VB.NET):**

Me.LimitOrder(4, OrderSide.Sell, 0, Data1)

Se obtiene es la cantidad de títulos/contratos ofertados en la cuarta posición de venta de la fuente Data1.

## LimitPrice

**Descripción:**
La propiedad permite obtener el precio de las órdenes activas de oferta y demanda para una posición concreta en una barra determinada.

Precaución. Sólo devuelve resultados durante el tiempo real, pues no existe histórico de las posiciones de oferta y demanda.

**Sintaxis:**
LimitePrice(Level, Side, BarsAgo, Identifier)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Level | 1 | Indica la n-ésima posición de oferta o demanda de la que deseamos saber el precio. |
| Side | OrderSide.Buy | Posición de la orden . Dato de tipo Type.OrderSide, pudiendo ser:<br>**OrderSide.Buy** (demanda)<br>**OrderSide.Sell** (oferta) |
| BarsAgo | 0 | Posición de la barra que se quiere consultar, aunque debe ser una barra generada durante el tiempo real. |
| Identifier | Data | Serie de datos de la que se extrae la información. |

**Ejemplo (VB.NET):**

Me.LimitOrder(4, osSell, 0, Data1) Devuelve el precio ofertado en la cuarta posición de venta del Data1.

## LimitVol

**Descripción:**
La propiedad LimitVol permite obtener el volumen de contratos existente de órdenes activas de oferta y demanda para una posición concreta en una barra determinada.

Esta propiedad sólo devuelve resultados durante el tiempo real, pues no existe histórico de las posiciones de oferta y demanda.

**Sintaxis:**
LimiteVol(Level, Side, BarsAgo, Identifier)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Level | 1 | Indica la n-ésima posición de oferta o demanda de la que deseamos saber la cantidad ofertada. |
| Side | OrderSide.Buy | Posición de la orden . Dato de tipo Type.OrderSide, pudiendo ser: |

| | | |
|---|---|---|
| | | **OrderSide.Buy** (demanda)<br>**OrderSide.Sell** (oferta) |
| BarsAgo | 0 | Posición de la barra que se quiere consultar, aunque debe ser una barra generada durante el tiempo real. |
| Identifier | Data | Dato del que se extrae la información. |

**Ejemplo (VB.NET):**

Me.LimitVol(2, OrderSide.Buy, 0, Data)

Se obtiene el volumen ofertado de contratos/títulos en la segunda posición de compra de la fuente Data.

## Low

**Descripción:**
Esta función devuelve el valor del mínimo de una barra para una serie de datos concreta.

**Sintaxis:**
Me.Identifier.Low(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hacer referencia a la barra actual. En este parámetro se puede utilizar cualquier valor numérico contenido en una variable o introducir directamente el valor. Puede especificarse como función sustituyendo el valor numérico. |

**Ejemplo (VB.NET):**

Me.Data2.Low(0) Retornará el mínimo de la barra actual (de la fuente de datos Data2).

## MarketFilledOrders

**Descripción:**
Esta función permite tener acceso a las órdenes ejecutadas de intermediación (órdenes del usuario) desde el propio sistema, por tanto, sólo tendrá sentido se ha establecido la conexión con el broker (simulada o real).

**Sintaxis:**
MarketFilledOrders(Label, BarsAgo, Side)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Label | - | Etiqueta asociada a la orden que se desea consultar. |
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hacer referencia a la barra actual. |
| Side | - | Posición de la orden a consultar. (0 posición de compra y 1 posición de venta). |

**Ejemplo (VB.NET):**

Supongamos que la estrategia “A” está comprada y coloca 2 órdenes a la vez (un objetivo de beneficios y un stop a mercado).

Una vez que se ha activado el trading de la estrategia, podemos saber desde el propio código, qué orden se ejecutó haciendo lo siguiente:

```
Dim MKFO() As MktFilledOrder

MKFO = Me.MarketFilledOrders("A", 0, osSell)     Solicitamos si se han ejecutado órdenes con etiqueta “A”
                                                 en la barra actual de venta.
n = UBound(MKFO)

If Err.Number = 0 Then                           Si se confirma que MKFO no está vacío, significa que alguna orden
                                                 se ha ejecutado.
   If UBound(MKFO) <> -1 Then
      PrecioOrdenSalida = MKFO(0).Price          Cogemos el precio que nos ha devuelto la función
   End If

End If
```

Según sea el valor de **PrecioOrdenSalida** tendremos una idea de si se ejecutó el stop o el objetivo.

## MinutesToTime

**Descripción:**

Es utilizada para transformar minutos en tiempo standard.

**Sintaxis:**

MinutesToTime(Minutes)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Minutes | - | Tiempo en formato numérico |

**Ejemplo (VB.NET):**

```
Me.MinutesToTime(300)          Retornará el valor 5 (300 minutos equivalen a 5 horas).
```

## NetProfit

**Descripción:**

Esta función se utiliza para obtener el valor de la ganancia neta que nuestro sistema lleva acumulada hasta la barra actual (barra en la que se están realizando los cálculos). Para obtener la ganancia total consideramos que el último negocio abierto concluye en el cierre de la barra actual.

**Sintaxis:**
NetProfit(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

```
Me.NetProfit(SttRepresentation.ByPoints)
```

Retornará en puntos la ganancia neta obtenida por la estrategia.

## NumberOfLines

**Descripción:**
Esta función devuelve el número de líneas que tiene el indicador sobre el que se aplica. El indicador debe haber asignado valores a las líneas para que el valor devuelto las incluya (solo devuelve el número de líneas que actualmente existen).

**Sintaxis:**
NumberOfLines

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Supongamos que creamos un indicador llamado MiIndicador que realiza el siguiente cálculo:

```
Valor1 = Me.Data.High + Me.Data.Low + Me.Data.Close / 3
Valor2 = Me.Data.High + Me.Data.Low / 2
Valor3 = Me.Data.Open + Me.Data.Close + Me.Data.Close(1) / 3
```

Y luego pintamos:

```
Me.SetIndicatorValue(Valor1, 1)
Me.SetIndicatorValue(Valor2, 2)
Me.SetIndicatorValue(Valor3, 3)
```

Si dentro de nuestro código utilizamos ésta función:

```
Dim nlineas As Integer =Me.NumberOfLines
```

nlineas será igual a 3 que es el número de líneas que hemos pintado.

## NumberOfLosingTrades

**Descripción:**
Devuelve el número de negocios perdedores realizados hasta ese momento. Este valor irá cambiando a medida que se generen nuevas barras y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
NumberOfLosingTrades

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Supongamos que deseamos saber, en una barra determinada, el número de negocios perdedores acumulados. Podriamos definir una variable:

```
Dim perdedores As  Long = Me.NumberOfLosingTrades
```

Devuelve el número de negocios perdedores acumulado hasta el momento en que se realiza la llamada a la propiedad.

## NumberOfTrades

**Descripción:**
Devuelve el número de negocios realizados hasta ese momento. Este valor irá cambiando a medida que se generen nuevas barras y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
NumberOfTrades

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Supongamos que deseamos saber, en una barra determinada, el número de negocios totales acumulados. Podriamos definir una variable:

```
Dim totalnegocios As Long =Me.NumberOfTrades
```

Devuelve el número de negocios acumulado hasta el momento en que se realiza la llamada a la propiedad.

## NumberOfWinningTrades

**Descripción:**
Devuelve el número de negocios ganadores realizados hasta ese momento. Este valor irá cambiando a medida que se generen nuevas barras y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
NumberOfWinningTrades

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Supongamos que deseamos saber, en una barra determinada, el número de negocios ganadores acumulados. Podriamos definir una variable:

Dim ganadores As Long =Me.NumberOfWinningTrades

Devuelve el número de negocios ganadores acumulado hasta el momento en que se realiza la llamada a la propiedad.

## Open

**Descripción:**
Esta función devuelve el valor de la apertura de una barra.

**Sintaxis:**
Me.Identifier.Open(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hace referencia a la barra actual. En este parámetro es posible indicar un valor numérico contenido en una variable o teclear directamente el valor.<br>Puede especificarse como función sustituyendo el valor numérico.<br>Este parámetro sólo permite valores positivos. |

**Ejemplo (VB.NET):**

Me.Data.Open(3)  Devuelve la apertura de 3 barras atrás de la fuente de datos principal (Data1).

## OpInt

**Descripción:**
Esta función devuelve el valor del **OpenInterest** de una barra para una serie de datos concreta

Precaución. Esta función sólo retorna resultados en el caso de contratos de futuros, donde se facilita esta información.

**Sintaxis:**
Me.future.OpInt(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. El valor por defecto hace referencia a la barra actual. Puede especificarse como función sustituyendo el valor numérico. Este parámetro sólo permite valores positivos. |

**Ejemplo (VB.NET):**

Me.Data.OpInt(5)

Devuelve el OpenInteres de hace 5 barras (de la serie de datos Data).

## PaintBar

**Descripción:**
Mediante esta opción,  es posible pintar barras en el gráfico con los valores deseados para apertura, máxima, mínima, cierre.

**Sintaxis:**
PaintBar(Open, High, Low, Close, Color, [LineNumber], [Width], [nBars])

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Open | - | Apertura de la barra. Permite utilizar una función o variable. |
| High | - | Máxima de la barra. Permite utilizar una función o variable. |
| Low | - | Mínima de la barra. Permite utilizar una función o variable. |
| Close | - | Cierre de la barra. Permite utilizar una función o variable. |
| Color | - | Color que se utilizará para pintar las barras. Se utiliza el tipo de dato **System.Drawing.Color.** |
| LineNumber | - | Especifica el número de orden de pintar estudio. Esto es para estudios que utilizan diferentes órdenes de pintar (p.e si queremos incluir en un |

| | | |
|---|---|---|
| | | estudio una orden de pintar tipo PintarBarra (PaintBar) y otra del tipo Pintar Línea (PaintSeries), debemos poner a la primera orden, número de línea 0 y a la segunda, número de línea 1. |
| Width | 1 | Grosor de la barra que se pintará. |
| nBarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

Queremos crear un gráfico basado en medias. Para ello, declaramos los siguiente cuatro objetos y el parámetro *Periodo* en un estudio nuevo:

```
<Parameter(Name:="Periodo", DefaultValue:=15, MinValue:=2, MaxValue:=100, Step:=1)>
Private period As Integer

Dim opendata As AvSimple
Dim closedata As AvSimple
Dim highdata As AvSimple
Dim lowdata As AvSimple
```

Seguidamente los creamos asignando a cada uno un campo de precio distinto:

```
Me.opendata = New AvSimple(Me.Data, Me.period, Price.Open)
Me.closedata = New AvSimple(Me.Data, Me.period, Price.Close)
Me.highdata = New AvSimple(Me.Data, Me.period, Price.High)
Me.lowdata = New AvSimple(Me.Data, Me.period, Price.Low)
```

Por ultimo, desde el método *OnCalculateBar* usamos el método PaintBar:

```
If (Me.closedata.Value() > Me.opendata.Value()) Then
    Me.PaintBar(Me.opendata.Value(), Me.highdata.Value(), Me.lowdata.Value(), Me.closedata.Value(),
    Color.Blue)
Else
    Me.PaintBar(Me.opendata.Value(), Me.highdata.Value(), Me.lowdata.Value(), Me.closedata.Value(),
    Color.Pink)
End If
```

Pintando las barras de azul o rosa según sea la tendencia.

## PaintCandlestick

**Descripción:**

Mediante esta opción, es posible pintar velas con los valores deseados para apertura, máxima, mínima y cierre.

**Sintaxis:**

PaintCandlestick(Open, High, Low, Close, Color, [LineNumber], [Width], [nBars])

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Open | - | Apertura de la barra. Permite utilizar una función o variable. |
| High | - | Máxima de la barra. Permite utilizar una función o variable. |
| Low | - | Mínima de la barra. Permite utilizar una función o variable. |
| Close | - | Cierre de la barra. Permite utilizar una función o variable. |
| Color | - | Color que se utilizará para pintar las barras. Se utiliza el tipo de dato **System.Drawing.Color**. |
| LineNumber | - | Especifica el número de orden de pintar estudio. Esto es para estudios que utilizan diferentes órdenes de pintar (p.e si queremos incluir en un estudio una orden de pintar tipo PintarBarra (.PaintBar) y otra del tipo Pintar Línea (.PaintSeries), debemos poner a la primera orden, número de línea 0 y a la segunda, número de línea 1. |
| Width | 1 | Grosor que se utilizará para pintar la vela. |
| nBarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

Queremos pintar velas usando los métodos GetTrueHigh y GetTrueLow. De modo que el máximo de cada vela será el valor devuelto por la función GetTrueHigh y el mínimo por la función GetTrueLow:

```
If (Me.Data.Close() > Me.Data.Open()) Then
    Me.PaintCandlestick(Me.Data.Open, Me.Data.GetTrueHigh, Me.Data.GetTrueLow, Me.Data.Close,
    Color.WhiteSmoke)
Else
    Me.PaintCandlestick(Me.Data.Open, Me.Data.GetTrueHigh, Me.Data.GetTrueLow, Me.Data.Close,
    Color.DarkGray)
End If
```

De modo que las velas serán de color humo si son alcistas y gris oscuro si son bajistas.

## PaintMaxMin

**Descripción:**

Esta función es similar a PaintBar y PaintCandlestick. La diferencia es que en ésta solo podremos establecer valores para la máxima y para la mínima de la barra que deseamos pintar.

**Sintaxis:**
PaintMaxMin(Top, Bottom, Color, [LineNumber], [Width], [nBars])

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Top | - | Máxima de la barra que vamos a pintar. Permite usar una función o variable. |
| Bottom | - | Mínima de la barra. Permite utilizar una función o variable. |
| Color | - | Color que se utilizará para pintar las barras. Se utiliza el tipo de dato **System.Drawing.Color.** |
| LineNumber | - | Especifica el número de orden de pintar estudio. Esto es para estudios que utilizan diferentes órdenes de pintar (p.e si queremos incluir en un estudio una orden de pintar tipo PintarBarra (.PaintBar) y otra del tipo Pintar Línea (.PaintSeries), debemos poner a la primera orden, número de línea 0 y a la segunda, número de línea 1. |
| Width | 1 | Grosor que se utilizará para pintar. |
| nBarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

Me.PaintMaxMin(Me.Data.Low(), Me.Data.Low(), Color.Blue, 0, 8, 0)

Pinta en el minimo de la barra un círculo (grosor 8) de color azul.

## PaintSeries

**Descripción:**

Esta función se utiliza para pintar líneas de datos en pantalla. Esta tarea también se puede realizar haciendo un indicador, pero en el supuesto de que queramos mezclar líneas con opciones de pintar barras, o pintar figuras, podemos optar por crear un estudio que incluya todos estos tipos. Hay que recordar aquí que los estudios no pueden ser utilizados en otro tipo de proyectos como estrategias o indicadores. Por tanto, si deseamos usar después la línea de datos debemos crear un indicador.
**Sintaxis:**
PaintSeries(Price, Color,[ LineNumber], [Width], [nBars])

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Price | - | En este parámetro se indica el valor que |

| | | |
|---|---|---|
| | | deseamos que tenga la línea en la barra actual. Cualquier valor numérico o una variable de tipo numérico son aceptados en este parámetro. |
| Color | - | Color que se utilizará para pintar las barras. Se utiliza el tipo de dato **System.Drawing.Color.** |
| LineNumber | - | Especifica el número de orden de pintar estudio. Esto es para estudios que utilizan diferentes órdenes de pintar (p.e si queremos incluir en un estudio una orden de pintar tipo PintarBarra (.PaintBar) y otra del tipo Pintar Línea (.PaintSeries), debemos poner a la primera orden, número de línea 0 y a la segunda, número de línea 1. |
| Width | 1 | Grosor que se utilizará para pintar la vela. |
| nBarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

PaintSeries( (Me.Data.High + Me.Data.Low) / 2, Color.Red, 0, 1, 0)

Dibuja una línea de color rojo que va representando el punto medio de la barra.

## PercentProfitable

**Descripción:**
Devuelve el ratio de fiabilidad. Este valor irá cambiando a medida que se generen nuevas barras y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
PercentProfitable

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Queremos saber en un momento determinado, el ratio de fiabilidad de nuestro sistema. Para esto, previamente definiaríamos una variable:

Dim Fiabilidad As Double = 0

A ésta, en un momento dado, se le asignará el valor de la propiedad .PercentProfitable:

Fiabilidad =Me.PercentProfitable

## ProfitFactor

**Descripción:**
Devuelve el ratio **Factor de ganancia**. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
ProfitFactor(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Queremos saber en un momento determinado, el factor de ganancia (porcentual) de nuestro sistema. Para esto, previamente definiaríamos una variable:

```
Dim fganancia As Double = 0
```

A ésta, en un momento dado, se le asignará el valor de esta función:

```
fganancia =Me.ProfitFactor(SttRepresentation.Porcentual)
```

## PRR

**Descripción:**
Devuelve el ratio **Factor de ganancia ajustada**. Este valor irá cambiando a medida que se generen nuevas barras, y su valor dependerá de la barra en la que se haga la llamada a dicha propiedad.

**Sintaxis:**
PRR(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Queremos saber en un momento determinado, el factor de ganancia ajustado (en puntos) de nuestro sistema. Para esto, previamente definiaríamos una variable:

```
Dim fganajus As Double = 0
```

A ésta, en un momento dado, se le asignará el valor de esta función:

fganajus =Me.PRR(SttRepresentation.ByPoints)

## RegressionAngle

**Descripción:**
Devuelve el valor del ángulo que forma con la horizontal la recta de regresión formada por los precios de cierre situados entre las barras **StartBar** y **EndBar**.

**Sintaxis:**
Me.Identifier.RegressionAngle(StarBar, EndBar, Data)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| StarBar | - | Barra de inicio de la recta de regresión. |
| EndBar | - | Barra de inicio de la recta de regresión. |

## RegressionSlope

**Descripción:**
Devuelve el valor de la pendiente de  la recta de regresión formada por los precios de cierre situados entre las barras **StartBar** y **EndBar**.

**Sintaxis:**
Me.Identifier.RegressionSlope(StarBar, EndBar)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| StarBar | - | Barra de inicio de la recta de regresión. |
| EndBar | - | Barra de inicio de la recta de regresión. |

## ReleaseDataIdentifier - RDI

**Descripción:**
Permite liberar un **DataIndentifier** que haya sido llamado previamente mediante el método *GetSymbolIndentifier*.

**Sintaxis:**
ReleaseDataIdentifier(Identifier)

También se puede utilizar el modo abreviado **RDI**.

RDI(Identfier)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Identifier | Data | Data que deseamos liberar. |

**Ejemplo (VB.NET):**

Primero, creamos un data identifier que vamos a usar para extraer el minimo movimiento del vencimiento FDAXU5 (contrato de septiembre-2015 del Futuro Dax)

```
Dim auxdata As DataIdentifer = Me.GetSymbolIdentifier("010015FDAXU5", 1, compresionRange.Dias, CDate("01/09/2015"), CDate("01/01/2036"))
```

Luego extraemos el minimo movimiento que vamos a usar:

```
Dim pip As Double = Me.auxdata.GetSymbolInfo(SymbolInfo.MinMov)
```

Por último, como el nuevodata está ocupando espacio en memoria, lo liberamos pues no vamos a usarlo más. Para ello usamos la función *ReleaseDataIdentifier*

```
Me.ReleaseDataIdentifier (auxdata)
```

## Sell

**Descripción:**
Es utilizada indistintamente para venta de futuros y venta a crédito de acciones. Es importante tener en cuenta que esta función se utiliza para abrir posiciones de venta, no para cerrar las de compra solamente. Por tanto, si deseamos cerrar una compra sin abrir una posición de venta a crédito, debemos usar la función ExitLong.

**Sintaxis:**
Sell(TradeType, Contracts, Price, Label)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Type | TradeType.AtClose | Tipo de orden que se desea lanzar (TraderType.AtClose, TraderType.AtMarket, TraderType.AtLimit y TraderType.AtStop). |
| Contracts | 1 | Número de contratos/acciones. Las especificaciones numéricas sobre contratos pueden ser sustituidas por variables o por cualquier función definida previamente. |
| Price | - | Precio de venta. Sólo se indica este parámetro en las órdenes del tipo **TraderType.AtStop** y **TraderType.AtLimit**. El valor puede expresarse mediante un número, una variable, una función, o bien una mezcla de valor y función. |
| Label | - | Etiqueta de la orden en formato texto. |

**Ejemplo (VB.NET):**

```
If (Me.GetMarketPosition() <> -1) then
    Me.Sell(TradeType.AtStop, 1, Me.Data.Close()-10, "V1")
```

End If

Lanza una orden de venta en stop (un contrato), siendo el precio del stop el cierre de la barra menos 10 puntos, y la etiqueta identificativa de la orden “V1”.

## SetBackGroundColor

**Descripción:**
Pinta el fondo de la ventana, de una barra determinada, en el color indicado.

**Sintaxis:**
SetBackGrounColor (BarsAgo, Color)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Color | - | Color en el que se debe pintar el fondo de la ventana en la barra indicada (BarsAgo). Utiliza el tipo **System.Drawing.Color.** |

**Ejemplo (VB.NET):**

Me.SetBackGroundColor(1, Color.Red) Pintará el fondo de la ventana (en la barra anterior) en color rojo.

## SetBarColor

**Descripción:**
Asigna a la barra indicada, de una línea concreta del indicador, un color determinado.

**Sintaxis:**
SetBarColor (BarsAgo, Line, Color)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| Color | - | Color en el que se debe pintar la barra indicada. Utiliza el tipo **System.Drawing.Color**. |

**Ejemplo (VB.NET):**

Me.SetBarColor(0,1, Color.Red) Pintará de color rojo la barra actual de la línea 1.

## SetBarProperties

**Descripción:**
Asigna a la barra indicada, de una línea concreta del indicador, el color, grosor y tipo de de línea y representación indicados en el resto de parámetros.

**Sintaxis:**
SetBarProperties (BarsAgo, Line, Color, Width, Style, Representation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | - | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| Color | - | Color en el que se debe pintar la barra indicada. El valor aplicado es del tipo **System.Drawing.Color**. |
| Width | - | Indica el grosor que se aplica a la barra (1,2,..). |
| Style | - | Estilo que se utilizará para la representación:<br>**LineStyle.Solid** línea continua<br>**LineStyle.Dash** línea discontinua<br>**LineStyle.Dot** línea punteada<br>**LineStyle.Dashdot** línea discontinua con punto<br>**LineStyle.Dashdotdot** línea discontinua con 2 puntos |
| Representation | - | Tipo de representación que se utilizará:<br>**IndicatorRepresentation.Bars** barras<br>**IndicatorRepresentation.Candlestic** velas<br>**IndicatorRepresentation.DottedLine** línea punteada<br>**IndicatorRepresentation.FilledHistogram** histograma relleno<br>**IndicatorRepresentation.Histogram** histograma<br>**IndicatorRepresentation.Lineal** lineal<br>**IndicatorRepresentation.Parabolic** parabólico<br>**IndicatorRepresentation.Volume** volumen |

**Ejemplo (VB.NET):**

```
Me.SetBarProperties(0,1, Color.Red,2,LineStyle.Solid,IndicatorRepresentation.Lineal)
```

Pintará en color rojo la línea 1 de la barra actual (en formato de línea continua y grosor 2).

**Novedades en Visual Chart 6:**

Como Visual Chart 6 permite utilizar las capacidades de .NET Framework, es posible crear atributos personalizados a fin de proporcionar información adicional sobre los elementos del programa. Como cada proyecto de indicador consiste en la creación de una clase, podemos incluir varios atributos a la clase en cuestión. Entre ellos, el atributo *OutputSeriesProperties*. Este atributo permite definir el estilo de las líneas del indicador de forma alternativa a la función *SetBarProperties*.

**Sintaxis:**

<OutputSeriesProperties(Line:=NumLine, Color:=Chart.Colors, ChartingStyle:=Chart.ChartStyle, Width:=WidthLine)>

El atributo *OutputSeriesProperties* se define debajo del atributo *Indicator*.

## SetBarRepresentation

**Descripción:**
Esta función se encarga de establecer el tipo de representación a una barra especificada.

**Sintaxis:**
SetBarRepresentation(BarsAgo, Line, Representation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| Representation | - | Tipo de representación que se utilizará, siendo del tipo **IndicatorRepresentation** (pueden consultarse los diferentes tipos en la función SetBarProperties). |

**Ejemplo (VB.NET):**

```
Me.SetBarRepresentation(0,1, IndicatorRepresentation.Lineal)
```

La barra actual de la línea 1 del indicador, se representará en formato lineal.

## SetBarStyle

**Descripción:**
Esta función se encarga de establecer el estilo a una barra especificada.

**Sintaxis:**
SetBarStyle(BarsAgo, Line, LineStyle)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| Representation | - | Estilo de línea que se utilizará, siendo del tipo **LineStyle** (pueden consultarse los diferentes tipos en la función SetBarProperties). |

**Ejemplo (VB.NET):**

Me.SetBarStyle(0,1,LineStyle.Solid)

La barra actual de la línea 1 del indicador, se representará en formato de línea continua.

## SetBarWidth

**Descripción:**
Asigna a la barra indicada, de una línea determinada, el grosor deseado.

**Sintaxis:**
SetBarWidth(BarsAgo, Line, Width)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| Width | - | Grosor en el que se representará la barra. |

**Ejemplo (VB.NET):**

Me.SetBarWidth(0,1, 2) La barra actual de la línea 1 del indicador, se representará con grosor 2.

## SetHistogramBand

**Descripción:**
Asigna a la barra indicada, de una línea de datos concreta, el valor que tiene la línea del histograma, es decir, el valor hasta el cual se pinta el histograma (si la representación utilizada es histograma). Esta función está estrictamente asociada al uso de la propiedad SetBarRepresentation.

**Sintaxis:**
SetHistogramBand(BarsAgo, Line, BandLine)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| BandLine | - | Línea de datos que servirá de referencia para dibujar el histograma. |

**Ejemplo (VB.NET):**

Supongamos que utilizamos la función .SetBarRepresentation(0,1, IndicatorRepresentation.Histogram) . El indicador necesitará un valor de referencia sobre el que oscilar para generar el histograma.

En este caso:

Me.SetIndicatorValue (Me.Data.Close – Me.Data.Close(1),1,0) Con la línea 1 se representará la diferencia entre cierres
Me.SetIndicatorValue(0,2,0) Con la línea 2 se representará el valor 0.

A continuación utilizando la función SetBarRepresentation, se indicará que la línea 1 se pintará por defecto con un histograma:

Me.SetBarRepresentation (0,1, IndicatorRepresentation.Histogram)

Y por último se indicará la línea de referencia sobre la que oscilar:

Me.SetHistogramBand (1,2)

De esta forma la línea 1 usará como línea de banda (línea de referencia) la línea 2.

## SetIndicatorPos

**Descripción:**
Indica a la barra indicada, de una línea determinada, una posición concreta.

**Sintaxis:**
SetIndicatorPos(BarsAgo, Line, IndicatorPosition)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Line | - | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| Indicator Position | - | La tendencia es del tipo IndicatorPosition, pudiendo ser **IndicatorPosition.Bull** (alcista), **IndicatorPosition.Bear** (bajista) o **IndicatorPosition.Neutral** (neutra). |

**Ejemplo (VB.NET):**

Me.SetIndicatorPos(3,2,ipNeutral) Asigna a la barra 3 hacia atrás, de la línea 2, la posición neutra.

## SetIndicatorValue

**Descripción:**
Método para asignar un valor del indicador en una barra indicada. A este valor se le puede asignar una determinada tendencia si especificamos algo en la propiedad *IndicatorPosition*.

**Novedades en Visual Chart 6:**

Si al pintar una línea x el valor de la barra indicada es relativo a una barra anterior (es decir, no especificamos 0 en la propiedad *BarsAgo*), debemos saber que entonces dicha línea quedará *bloqueada* para

su uso desde una estrategia. Esto se hace para evitar que las estrategias estén basadas en indicadores que proyectan la información hacia atrás.

**Sintaxis:**
SetIndicatorPos(Value, Line, BarsAgo, IndicatorPosition)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Value | - | Valor numérico de salida. |
| Line | 1 | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| BarsAgo | - | Número de barras hacia atrás. El valor 0 hace referencia a la barra actual. |
| Indicator Position | - | La tendencia es del tipo IndicatorPosition, pudiendo ser **IndicatorPosition.Bull** (alcista), **IndicatorPosition.Bear** (bajista) o **IndicatorPosition.Neutral** (neutra). |

### Ejemplo (VB.NET):

Supongamos que definimos una variable numérica y a ésta le asignamos el valor que retorna el siguiente cálculo:

```
Dim buffer3 As Double  = Me.Data.High(0) – Me.Data.Low(0) / Me.Data.High(0) * 100
```

A continuación, podemos utilizar la función *SetIndicatorValue* para pintar en cada barra el valor calculado para dicha variable:

```
If (buffer3 > 50) then
    Me.SetIndicatorValue(buffer3, 1,0,IndicatorPosition.Bull)
Else
    Me.SetIndicatorValue(buffer3, 1,0,IndicatorPosition.Bear)
End If
```

La diferencia está en que dependiendo de cómo sea este valor, se le asignará al indicador tendencia alcista o bajista.

## SetLineName

**Descripción:**
Asigna un nombre a la línea indicada. Este método no es necesario especificarlo una vez por barra, ya que en el momento en que una línea lleva asignado un nombre, es válido para todas las barras. Por tanto, aconsejamos declarar la función *SetLineName* desde el método *OnInitCalculate.*

**Sintaxis:**
SetLineName(Line, LineName)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|

| | | |
|---|---|---|
| Line | - | Identifica la línea de datos a la que pertenece la barra sobre la que se establecerá la propiedad. |
| LineName | - | El nombre que se le asigna a la línea correspondiente. |

**Ejemplo (VB.NET):**

Desde el método *OnInitCalculate* especificamos lo siguiente:

Me.SetLineName(2, “DT”)  De modo que se asigna el nombre “DT” a la línea 2.

**Novedades en Visual Chart 6:**

Como Visual Chart 6 permite utilizar las capacidades de .NET Framework, es posible crear atributos personalizados a fin de proporcionar información adicional sobre los elementos del programa. Como cada proyecto de indicador consiste en la creación de una clase, podemos incluir varios atributos a la clase en cuestión. Entre ellos, el atributo *OutputSeriesProperties*. Este atributo permite definir el nombre de las líneas del indicador de forma alternativa al uso de la función *SetLineName*.

**Sintaxis:**
<OutputSeriesProperties(Name:=LineName)>

El atributo *OutputSeriesProperties* se define debajo del atributo *Indicator*.

## SetWndBackGroundColor

**Descripción:**
Asigna el color de fondo de la ventana de un indicador.

**Sintaxis:**
SetWndBackGrounColor()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Color | - | Valor del tipo System.Drawing.Color para indicar el color de fondo de la ventana. |

**Ejemplo (VB.NET):**

Me.SetWndBackgroundColor(Color.Beige)  Asigna el color beis a la ventana del indicador.

## ShouldTerminated

**Descripción:**
Se utiliza para interrumpir los cálculos de una estrategia. **ShouldTerminated** es una variable booleana inicializada con el valor False, de forma que para la interrupción de los cálculos,  se tiene que asignar el valor True a la misma.

Resulta útil cuando se trabaja con históricos largos y se desea detener el cálculo de una estrategia cuando determinadas circunstancias se produzcan.

**Sintaxis:**
ShouldTerminated = True/False

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

```
If Me.CurrentBar = 1000 And Me.NetProfit < 100 Then
    Me.ShouldTerminate = True
End If
```

En este caso, los cálculos se detendrán cuando hayan transcurrido 1000 barras de cálculo y la ganancia neta sea inferior a 100 euros.

## Slope

**Descripción:**
Devuelve el valor de la pendiente de  la recta de regresión, formada por los precios indicados, situados entre las barras **StartBar** y **EndBar** para una serie de datos determinada.

La función Slope devuelve para cada barra el valor de la pendiente de la ecuación de regresión, que relaciona las cotizaciones con la variable tiempo, pudiendo interpretarse como un indicador de la pendiente de la tendencia.

**Sintaxis:**
Me.Identifier.Slope(StarBar, EndBar, StarPrice, EndPrice)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| StarBar | - | Número de barra del inicio de la recta de regresión. |
| EndBar | - | Número de barra del fin de la recta de regresión. |
| StarPrice | - | Precio de inicio de la recta de regresión. |
| EndPrice | - | Precio de fin de la recta de regresión. |

**Ejemplo:**
Queremos entrar a mercado cada vez que se produzca un cambio de pendiente de la recta de regresión. De modo que si la pendiente pasa de positiva a negativa entraremos cortos y si pasa de negativa a positiva a largos.
Lo primero, extraemos la pendiente de la recta de las últimas 10 barras en la barra actual y la misma pendiente en la barra anterior:

```
Dim slope As Double = Me.avdata.Slope(Bar - 10, Bar, Me.avdata.Value(10), Me.avdata.Value(0))
Dim slope_ant As Double = Me.avdata.Slope(Bar - 11, Bar - 1, Me.avdata.Value(11), Me.avdata.Value(1))
```

Despúes usamos esa información para definir las reglas de operativa:

```
If (slope > 0 And slope_ant <= 0) Then
    Me.Buy(TradeType.AtMarket)
```

```
ElseIf (slope < 0 And slope_ant >= 0) Then
    Me.Sell(TradeType.AtMarket)
End If
```

## StandardDeviation

**Descripción:**
Retorna la desviación estándar.

**Sintaxis:**
StandardDeviation(Show As SttRepresentation)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | ByPoints | Permite indicar el formato en el que se mostrará la información, siendo del tipo **SttRepresentation**, que puede tomar los valores SttRepresentation.**ByPoints** (en puntos), o SttRepresentation.**Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

```
Dim DE as double = Me.StandardDeviation(SttRepresentation.ByPoints)
```

En el momento en que se hace uso de esta propiedad, se le asignará a la variable DE, la desviación standard en puntos.

## StartBar

**Descripción:**
Permite especificar a partir de qué barra se va a empezar a calcular la estrategia.

La asignación de la propiedad *StartBar* se realiza desde el método *OnInitCalculate*.

**Sintaxis:**
StarBar = (Número de barras)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | - |

**Ejemplo (VB.NET):**

Supongamos que nuestra estrategia opera en base a ciertos niveles de soporte y resistencia calculados en base a las últimas 25 barras.

Como necesitamos que al menos haya 24 barras previas, especificamos que no empiece el proceso hasta la barra 25:

```
Public Overrides Sub OnInitCalculate()
    Me.StartBar = 25
End Sub
```

## Time

**Descripción**
Devuelve el valor de campo Tiempo (hora) de una barra. La hora de una barra viene dada por la hora de finalización del periodo temporal que resume la barra. La hora de una barra es considerada en formato militar (HHMM), es decir, si la hora de una barra es 5:35pm, en visual chart se considera como el valor numérico 17:35h.

**Sintaxis:**
Me.Identifier.Time(BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barras hacia atrás. Por defecto hace referencia a la barra actual. |

**Ejemplo (VB.NET):**

Me.Data.Time(3) Retorna la hora en formato militar (HHMM) de hace 3 barras en la fuente de datos Data1.

## TimeEx

**Descripción**
Retorna la fecha de la barra de referencia en formato fecha (DD/MM/AAAA HH:MM:SS).

**Sintaxis:**
Me.Identifier.TimeEx(TickIndex, BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| TickIndex | 0 | Parámetro de salida que es obligatorio incluir. Para ello, declararemos una variable donde almacenar el valor devuelto. Sólo tiene efecto en gráficos de tics. |
| BarsAgo | 0 | Barra de la que se desea extraer la fecha. El valor por defecto hace referencia a la barra actual. |

TickIndex es un parámetro de Salida. Cuando se opera sobre un gráfico de ticks, puede ocurrir que algunos de éstos tengan la misma fecha. Este parámetro se rellena indicando la n-ésima posición de ticks referente a la misma fecha.

**Ejemplo (VB.NET):**

Declaramos la variable de tipo Long donde almacenar el número de tic:
Dim tickindex As Long = 0

La función
Me.Data2.TimeExe(tickindex, 1)

Retorna la fecha (DD/MM/AAAA HH:MM:SS) de la barra anterior de la serie Data2.

## TimeToMinutes

**Descripción:**
Retorna el número de minutos transcurridos desde las 00:00 horas.

**Sintaxis:**
TimeToMinutes(Time)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Time | - | Hora en formato militar (HHMM) |

**Ejemplo (VB.NET):**

Me.TimeToMinutes(1735)  Retorna 1055 minutos.

## TodayCurrentBar

**Descripción:**
Devuelve el número de barra respecto al total de barras por sesión. Dicho de otro modo, devuelve la distancia en barras respecto a la barra de inicio de sesión.
**Novedades Visual Chart 6:**
Si trabajamos con estrategias *End Of Day* en las cuales aplicamos filtros horarios para determinar los intervalos de operativa, se recomienda que los parámetros que establecen los márgenes horarios vengan especificados por número de barras. Es decir, en lugar de especificar *InitTime* igual a 930 (por ejemplo), pasaríamos a definir este parámetro como *InitBar* igual a 2 (por ejemplo). Gracias a la nueva propiedad *TodayCurrentBar*, controlar si ha alcanzado o no a la barra correspondiente es sumamente sencillo.

**Sintaxis:**
TodayCurrentBar()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | |

**Ejemplo (VB.NET):**

Incluimos en una estrategia dos parámetros para limitar el periodo de operativa por sesión:

```
<Parameter(Name:="InitBar", DefaultValue:=2, MinValue:=1, MaxValue:=20, Step:=1)>
Private initbar As Integer
<Parameter(Name:="EndBar", DefaultValue:=40, MinValue:=25, MaxValue:=50, Step:=1)>
Private endbar As Integer
```

Luego especificamos que solo compruebe las reglas de entrada dentro de dicho intervalo:

```
If (Me.TodayCurrentBar >= Me.initbar And Me.TodayCurrentBar < Me.endbar) Then
   If (Me.rsidata.Value() < 70 And Me.rsidata.Value(1) >= 70) Then
      Me.Buy(TradeType.AtMarket)
   End If
End If
```

## TodayHigh

**Descripción:**
Devuelve el precio más alto dado dentro de la última sesión.

**Sintaxis:**
TodayHigh()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | |

**Ejemplo (VB.NET):**

```
If GetMarketPosition() = 1 Then
   Me.ExitLong(TradeType.AtLimit, 1, Me.TodayHigh())
End If
```

Activa una orden de salida limitada en base al precio máximo de la sesión.

## TodayLow

**Descripción:**
Devuelve el precio más bajo dado dentro de la última sesión.

**Sintaxis:**
TodayLow()

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| - | - | |

**Ejemplo (VB.NET):**

```
If GetMarketPosition() = -1 Then
   Me.ExitShort(TradeType.AtLimit, 1, Me.TodayLow())
End If
```

Activa una orden de cerrar cortos limitada en base al precio mínimo de la sesión.

## Volume

**Descripción:**
Devuelve el valor del volumen (acciones/contratos negociados) de una barra para una fuente de datos concreta.

**Sintaxis:**
Me.Identifier.Volume (BarsAgo)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| BarsAgo | 0 | Número de barra. El valor por defecto hace referencia a la barra actual. En este parámetro podemos indicar cualquier valor numérico contenido en una variable o teclear directamente el valor. Puede especificarse como función sustituyendo el valor numérico. |

**Ejemplo (VB.NET):**

Me.Data2.Volume(15)  Retorna el volumen  negociado 15 barras atrás, en la serie de datos Data2.

## WorstSeries

**Descripción:**
Devuelve la peor serie de negocios según sus resultados.

**Sintaxis:**
Worstseries(Show)

**Parámetros:**

| Nombre | Defecto | Descripción |
|---|---|---|
| Show | Bypoints | Permite indicar el formato en el que se mostrará la información, **SttRepresentation.ByPoints** (en puntos), o **SttRepresentation.Porcentual** (porcentaje). |

**Ejemplo (VB.NET):**

Dim peorserie As Double = Me.WorstSeries(SttRepresentation.ByPoints)

Asigna a la variable la peor serie en puntos hasta el momento actual.