MQR - Manual completo

Referencia práctica para el lenguaje MQR con enfoque en backtesting, eventos, órdenes, posiciones, bucles de gestión y exportación a MT5.

Responsable: Vinicius Fávero
Sitio web: rikdomchart.com.br

Contenido

---

1) Descripción general

MQR es un DSL para estrategias comerciales con una sintaxis inspirada en C++/MQL5. El flujo estándar es: escribir estrategia, ejecutar backtest, ajustar parámetros y, cuando sea necesario, exportar a MT5.

Punto clave: Incluso sin `OnBarOpenM1()`, el motor mantiene TP/SL y la precisión de ejecución pendiente en M1 cuando hay posiciones/órdenes abiertas en el modo OHLC de M1.

2) Archivos y canalización

Extensión	Objetivo
`.mqr`	Código fuente de la estrategia
`.exr`	Estrategia compilada para su ejecución por el motor.

FLUJO RECOMENDADO

```
1) Escribir la estrategia en MQR
2) Validarla en el backtest interno
3) Optimizar parámetros
4) Revalidar en M1 OHLC
5) Exportar a MT5 cuando tenga sentido
```

3) Programa y estructura del evento

Evento	Cuándo se activa	Uso común
`OnInit()`	Una vez al principio	SetTimeFrame, estado inicial, buffers
`OnBarOpenM1()`	Cerrando cada M1	Gestión, programación, seguimiento, limpieza.
`OnBarOpen()`	Primer tic después del límite TF	Señales de entrada y filtros.

Alias: `OnBarOpen()` es un alias de `OnBarOpen()` y `OnBarOpenM1()` es un alias de `OnBarOpenM1()`. `OnBar()` también funciona como alias para `OnBarOpen()`. Todos producen el mismo comportamiento.

Semántica compatible con MT5: `OnBarOpen()` se activa **primer tic de la siguiente vela** hasta el límite, como en MT5. `Close[0]` = formación de barras, `Close[1]` = barra completada. Para señales confirmadas, utilice el desplazamiento 1.

MODELO BASE

```
input ENUM_TIMEFRAMES inpTF = PERIOD_M5;

void OnInit() {
  SetTimeFrame(inpTF);
}

void OnBarOpenM1() {
  if (Hour() >= 17 && Minute() >= 30 && PositionsTotal() > 0) {
    CloseAll();
  }
}

void OnBarOpen() {
  if (BarIndex() < 30) return;
  if (PositionsTotal() > 0) return;

  // Close[1] = barra completa, EMA(21,1) = indicador de barra completa
  double ema = EMA(21, 1);
  if (Close[1] > ema) {
    // Close[0] = precio actual (barra formadora) para ejecución
    Buy(1.0, Close[0] - 80, Close[0] + 120);
  }
}
```

---

4) Tipos, entradas, enumeraciones y operadores

Tipos de bases

- `double`, `int`, `bool`
- `ENUM_TIMEFRAMES`
- `enum` personalizado

Operadores

- Aritmética: `+ - * / %`
- Comparación: `== != < > <= >=`
- Lógico: `&& || !`
- Asignación: `= += -= *= /= ++ --`

ENUMERACIÓN + ENTRADAS

```
enum ENUM_DIRECAO {
  COMPRA = 0,
  VENDA  = 1,
  AMBOS  = 2
};

input group "Riesgo"
input double TP = 120.0 [60, 10, 300];
input double SL = 90.0  [30, 10, 250];

input group "Ejecución"
input ENUM_DIRECAO Direcao = AMBOS;
input ENUM_TIMEFRAMES TF = PERIOD_M5;
```

---

5) Funciones Matemáticas

El lenguaje MQR ofrece funciones matemáticas integradas que operan con valores `double`. Se utilizan para redondeos, límites, potencias y cálculos generales.

Función	Parámetros	Devolver	Descripción
`Abs(x)`	x: double	double	Valor absoluto de x
`Max(x, y)`	x, y: double	double	Valor más grande entre x e y
`Min(x, y)`	x, y: double	double	Valor más pequeño entre x e y
`Sqrt(x)`	x: double	double	Raíz cuadrada de x
`Round(x)`	x: double	double	Redondear al entero más cercano
`Floor(x)`	x: double	double	Redondear hacia abajo (entero mayor ≤ x)
`Ceil(x)`	x: double	double	Redondear hacia arriba (entero más pequeño ≥ x)
`Pow(base, exp)`	base, exp: double	double	Potencia: base elevada a exp.

EJEMPLOS DE USO

```
// Redondeo de tiempo
int horaFechamento = Floor(inpHora / 100);
int minutoFechamento = inpHora % 100;

// Parada dinámica con ATR redondeado
double stopPts = Round(ATR(14, 0) * 1.5);

// Tamaño del rango en ticks
double rangeTicks = Round(Abs(High[0] - Low[0]) / TickSize());

// El máximo más alto de las últimas 3 velas.
double maxHigh = Max(High[0], Max(High[1], High[2]));

// Potencia para cálculos manuales de desviación estándar
double diff = Close[0] - SMA(20, 0);
double diffSq = Pow(diff, 2.0);

// Raíz cuadrada (por ejemplo, para normalización)
double vol = Sqrt(Pow(High[0] - Low[0], 2.0));
```

---

6) Funciones de fecha y hora

Funciones para consultar la hora, minuto y día de la semana de la vela M1 actual. Útil para filtros de horas de funcionamiento, cierres y control de sesiones.

Función	Devolver	Descripción
`Hour()`	entero	Tiempo actual de vela M1 (0–23)
`Minute()`	entero	Minuto de vela M1 actual (0–59)
`DayOfWeek()`	entero	Día de la semana (0=domingo, 1=lunes,..., 6=sábado)
`IsFirstBarOfDay()`	booleano	Verdadero si es la primera vela del día en el marco temporal principal
`IsLastBarOfDay()`	booleano	Verdadero si es la última vela del día en el marco temporal principal
`IsFirstBarOfDayTF(tf)`	booleano	Misma lógica, pero en el plazo `tf`
`IsLastBarOfDayTF(tf)`	booleano	Misma lógica, pero en el plazo `tf`

FILTRO DE HORA ESTÁNDAR

```
bool horarioPermitido() {
    int hAbertura = Floor(inpHoraAbertura / 100);
    int mAbertura = inpHoraAbertura % 100;
    int hEncerramento = Floor(inpHoraEncerramento / 100);
    int mEncerramento = inpHoraEncerramento % 100;

    bool aposInicio = (Hour() > hAbertura || (Hour() == hAbertura && Minute() >= mAbertura));
    bool antesEncerramento = (Hour() < hEncerramento || (Hour() == hEncerramento && Minute() <=
mEncerramento));

    return aposInicio && antesEncerramento;
}

void OnBarOpenM1() {
    // Todo cierra a las 5:30 pm
    if (PositionsTotal() > 0 && (Hour() > 17 || (Hour() == 17 && Minute() >= 30))) {
        CloseAll();
    }
}

void OnBarOpen() {
    // Restablecimiento del estado en la primera vela del día.
    if (IsFirstBarOfDay()) {
        // ... restablecer variables
    }
}
```

EJEMPLO PRÁCTICO DE VALIDACIÓN DE TIEMPOS: INICIO, LÍMITE DE ENTRADA Y CIERRE TOTAL

```
enum ENUM_HORA {
    H0905 = 0905,
    H1600 = 1600,
    H1630 = 1630,
    H1720 = 1720
  };

  input ENUM_HORA inpHoraInicio = H0905; // puede abrir nuevos pedidos
  input ENUM_HORA inpHoraLimite = H1630; // Después de eso, no se abren nuevos pedidos.
  input ENUM_HORA inpHoraFech   = H1720; // cierre total

  int HoraAtual() {
    return Hour() * 100 + Minute();
  }

  bool DentroDoHorarioOrdens() {
    int agora = HoraAtual();
    return (agora >= inpHoraInicio) && (agora < inpHoraLimite);
  }

  void FecharPorHorario() {
    if (HoraAtual() >= inpHoraFech) {
      if (PositionsTotal() > 0 || OrdersTotal() > 0) {
        CloseAll();
      }
    }
  }

  void OnBarOpenM1() {
    // Asegura el cierre en el momento exacto (granularidad M1)
    FecharPorHorario();
  }

  void OnBarOpen() {
    // También protege en el flujo principal.
    FecharPorHorario();

    // No hay nuevas entradas después de la hora límite
    if (!DentroDoHorarioOrdens()) return;

    // ... lógica de entrada
  }
```

7) Propiedades de activos, matrices y utilidades

7.1) Propiedades de los activos

Información actual de instrumentos financieros, útil para calcular paradas en puntos, normalizar precios y dimensionar lotes.

Función	Devolver	Descripción
`TickSize()`	double	Variación de precio más pequeña del activo (por ejemplo: 0,01 para miniíndice, 0,5 para minidólar)
`TickValue()`	double	Valor monetario de 1 tick por lote (ej: R$ 0,20 para WINFUT)
`Digits()`	entero	Número de decimales en el precio
`NormalizePrice(price)`	double	Redondea el precio al incremento válido del activo.

STOP BASADO EN TICKS Y PRECIO NORMALIZADO

```
void OnBarOpen() {
    if (PositionsTotal() > 0) return;

    double p = Close[0];
    double sl = NormalizePrice(p - 100 * TickSize());
    double tp = NormalizePrice(p + 200 * TickSize());

    Buy(1.0, sl, tp);
}
```

7.2) Matrices

Admite matrices dinámicas para almacenar series y colecciones de valores.

Función	Parámetros	Descripción
`ArrayResize(arr, size)`	arreglo: matriz, tamaño: int	Cambia el tamaño de la matriz al nuevo tamaño
`ArraySize(arr)`	arreglo: matriz	Devuelve el tamaño actual de la matriz.

EJEMPLO CON MATRIZ

```
double historico[];
int contagem = 0;

void OnBarOpen() {
    contagem++;
    ArrayResize(historico, contagem);
    historico[contagem - 1] = Close[0];

    if (ArraySize(historico) >= 10) {
        // utilizar los últimos 10 valores
    }
}
```

7.3) Imprimir (depurar)

Imprime valores en el registro de ejecución para depurarlos durante la prueba retrospectiva.

Función	Parámetros	Descripción
`Print(value)`	valor: double o string	Imprime en el registro de ejecución. Utilice la concatenación con + para redactar mensajes.

DEPURAR CON IMPRESIÓN

```
void OnBarOpen() {
    double rsi = RSI(14, 0);
    double atr = ATR(14, 0);
    Print("RSI=" + rsi + "ATR=" + atr + "Cerrar=" + Close[0]);
}
```

---

8) Plazos y Plazos Múltiples (MTF)

El plazo principal se define en `OnInit()` con `SetTimeFrame()`. Las lecturas de MTF utilizan funciones `*TF` sin perder el contexto de la vela principal.

8.1) Datos del período de tiempo de backtest (contexto principal)

Las series `Open[]`, `High[]`, `Low[]`, `Close[]`, `Volume[]` y `Time[]` siempre reflejan el marco temporal principal configurado en la estrategia. Este es el contexto donde `OnBarOpen()` toma decisiones de entrada.

Función/Serie	Descripción
`Open[n], High[n], Low[n], Close[n]`	OHLC del período de tiempo principal (n=0 formación de barras, n=1 última finalización)
`Volume[n], Time[n]`	Volumen de vela del contexto principal y marca de tiempo
`BarIndex(), BarCount()`	Índice actual y número de barras cargadas.

LECTURA DEL CONTEXTO PRINCIPAL

```
void OnBarOpen() {
  if (BarIndex() < 30) return;

  // Close[1] = barra completa, Close[2] = anterior a ella
  double c1 = Close[1];
  double c2 = Close[2];
  double rangeCompletada = High[1] - Low[1];

  // Ejemplo: aceleración simple en la barra completa
  if (c1 > c2 && rangeCompletada > TickSize() * 20) {
    Print("Señal en TF principal");
  }
}
```

8.2) Datos de otras franjas temporales

Para consultar el contexto externo (H1, H4, D1, etc.), use la familia `*TF`: `OpenTF`, `EMATF`, `RSITF`, etc. Esto evita reconstruir velas manualmente.

Categoría	Funciones
OHLC/volumen/tiempo	`OpenTF`, `HighTF`, `LowTF`, `VolumeTF`, `TimeTF`
Indicadores	`SMATF`, `EMATF`, `RSITF`, `ATRTF`, `MACDTF`, `MACDSignalTF`, `MACDHistTF`, `BollingerUpperTF`, `BollingerLowerTF`, `StochKTF`, `StochDTF`, `KeltnerUpperTF`, `KeltnerLowerTF`
Estado de la barra	`BarCountTF`, `IsBarCompleteTF`, `IsNewBarTF`

ENTRADA EN M5 CON FILTRO ESTRUCTURAL H1

```
void OnBarOpen() {
  // Contexto de backtest (M5, por ejemplo)
  double closeLocal = Close[0];

  // Contexto externo (H1)
  double openH1 = OpenTF(PERIOD_H1, 0);
  double emaH1 = EMATF(PERIOD_H1, 34, 0);

  if (openH1 > emaH1 && closeLocal > EMA(9, 0) && PositionsTotal() == 0) {
    Buy(1.0, closeLocal - 90, closeLocal + 180);
  }
}
```

8.3) Cómo trabajar con indicadores en el mismo código

Un enfoque práctico es separar: el contexto local como desencadenante y el contexto externo como dirección. Ejemplos comunes:

- Dirección macro en H1/H4 con `EMATF`.
- Momento de entrada al TF principal con `RSI` o cruzando los promedios locales.
- La volatilidad deja de dimensionarse con `ATR` (local) o `ATRTF` (externo).

COMBINACIÓN PRÁCTICA: TENDENCIA H1 + SINCRONIZACIÓN LOCAL + PARADA ATR

```
void OnBarOpen() {
  if (BarIndex() < 60 || PositionsTotal() > 0) return;

  double trend = EMATF(PERIOD_H1, 50, 0);
  double openH1 = OpenTF(PERIOD_H1, 0);
  double rsiLocal = RSI(14, 0);
  double atrLocal = ATR(14, 0);
  double p = Close[0];

  if (openH1 > trend && rsiLocal < 35) {
    Buy(1.0, p - atrLocal * 1.5, p + atrLocal * 3.0);
  }
}
```

8.4) Diferencia práctica: datos locales versus datos externos

Regla rápida: `Close[0]` es el **barra en formación** (barra de formación) del marco temporal de la estrategia: el mismo concepto que MT5. `Close[1]` es la última barra completada. `OpenTF(PERIOD_H1, 0)` devuelve la apertura de H1, incluso si su estrategia se ejecuta en M1/M5.

FILTRO DE TENDENCIA H1 EN ESTRATEGIA M5

```
void OnBarOpen() {
  double trendH1 = EMATF(PERIOD_H1, 34, 0);
  double openH1 = OpenTF(PERIOD_H1, 0);

  if (openH1 > trendH1 && RSI(14, 0) < 35 && PositionsTotal() == 0) {
    double p = Close[0];
    Buy(1.0, p - 90, p + 180);
  }
}
```

8.5) Confluencia del triple marco temporal (M5 + H1 + H4)

Ejemplo de entrada local en el timeframe backtest con doble confirmación externa. Este patrón reduce las señales en contra de la dirección macro.

CONFLUENCIA DE TENDENCIAS EN TRES HORIZONTES

```
void OnBarOpen() {
  if (PositionsTotal() > 0 || BarIndex() < 80) return;

  double p = Close[0];                  // TF de prueba retrospectiva
  double emaM5 = EMA(21, 0);            // indicador local
  double emaH1 = EMATF(PERIOD_H1, 34, 0);
  double openH1 = OpenTF(PERIOD_H1, 0);
  double emaH4 = EMATF(PERIOD_H4, 55, 0);
  double openH4 = OpenTF(PERIOD_H4, 0);

  bool macroAlta = (openH1 > emaH1) && (openH4 > emaH4);
  bool gatilhoLocal = p > emaM5 && RSI(14, 0) < 45;

  if (macroAlta && gatilhoLocal) {
    Buy(1.0, p - 100, p + 220);
  }
}
```

8.6) RSI local con RSI externo (evitar entrada de escape)

En este patrón, el disparo ocurre en el TF principal, pero el RSI de H1 valida si el movimiento externo ya está extendido.

RSI LOCAL + RSI H1

```
void OnBarOpen() {
  if (PositionsTotal() > 0 || BarIndex() < 50) return;

  double rsiM5 = RSI(14, 0);                    // ubicación
  double rsiH1 = RSITF(PERIOD_H1, 14, 0);       // externo
  double p = Close[0];

  // Lectura de ejemplo: compre solo con retroceso local y el primer semestre aún saludable
  if (rsiM5 < 35 && rsiH1 > 45 && rsiH1 < 70) {
    Buy(1.0, p - 90, p + 170);
  }
}
```

8.7) Sincronización por nueva barra de otro timeframe

Utilice `IsNewBarTF()` para actualizar las variables de contexto solo cuando se cierre la barra TF externa, evitando volver a calcular todo en cada vela local.

ACTUALIZACIÓN DEL CONTEXTO H1 SOLO EN EL CIERRE DEL H1

```
double contextoH1 = 0.0;

void OnBarOpenM1() {
  if (IsNewBarTF(PERIOD_H1)) {
    double oH1 = OpenTF(PERIOD_H1, 0);
    double eH1 = EMATF(PERIOD_H1, 34, 0);
    contextoH1 = oH1 - eH1; // positivo = por encima del promedio
  }
}

void OnBarOpen() {
  if (PositionsTotal() > 0) return;
  if (contextoH1 > 0 && RSI(14, 0) < 40) {
    double p = Close[0];
    Buy(1.0, p - 80, p + 160);
  }
}
```

8.8) ATR externo para parada dinámica

Cuando el TF principal es muy corto, el uso de ATR de H1/H4 puede estabilizar el tamaño de la parada en escenarios de ruido intradiario.

PARADA BASADA EN H1 ATR

```
input double FatorSL = 1.2;
input double FatorTP = 2.4;

void OnBarOpen() {
  if (PositionsTotal() > 0 || BarIndex() < 80) return;

  double atrH1 = ATRTF(PERIOD_H1, 14, 0);
  double p = Close[0];

  if (OpenTF(PERIOD_H1, 0) > EMATF(PERIOD_H1, 50, 0) && RSI(14, 0) < 38) {
    double sl = p - atrH1 * FatorSL;
    double tp = p + atrH1 * FatorTP;
    Buy(1.0, sl, tp);
  }
}
```

8.9) Ejemplo de lectura combinada de OHL en TF externo

Ejemplo simple de amplitud de vela en H1 utilizada como filtro para entrada local en el período de tiempo de backtest. Sólo utilizamos Open, High y Low del TF externo.

FILTRO DE AMPLITUD H1

```
bool CandleAmploH1() {
  double o = OpenTF(PERIOD_H1, 0);
  double h = HighTF(PERIOD_H1, 0);
  double l = LowTF(PERIOD_H1, 0);

  double range = h - l;
  if (range <= 0) return false;

  // La apertura en la mitad inferior sugiere presión de compra
  return (o - l) / range <= 0.4;
}

void OnBarOpen() {
  if (PositionsTotal() > 0) return;
  if (CandleAmploH1() && RSI(14, 0) < 45) {
    double p = Close[0];
    Buy(1.0, p - 85, p + 170);
  }
}
```

9) Indicadores Técnicos

El lenguaje MQR ofrece indicadores técnicos integrados con precálculo diferido y búsqueda O(1) por barra. Todos aceptan un parámetro opcional `offset` (predeterminado 0), donde 0 = barra formando, 1 = última barra completada, etc.

9.1) Indicadores Básicos

Función	Parámetros	Devolver	Descripción
`SMA(period, offset)`	periodo=número de barras, desplazamiento=desplazamiento (0=barra formando)	double	Media móvil simple
`EMA(period, offset)`	periodo=número de barras, desplazamiento=desplazamiento	double	Media móvil exponencial
`RSI(period, offset)`	periodo=número de barras, desplazamiento=desplazamiento	double (0–100)	Índice de fuerza relativa
`ATR(period, offset)`	periodo=número de barras, desplazamiento=desplazamiento	double	Rango verdadero promedio (volatilidad)

CRUCE MEDIO CLÁSICO

```
void OnBarOpen() {
    if (PositionsTotal() > 0 || BarIndex() < 30) return;

    // Barras completadas: desplazamiento 1 y 2
    double emaRapida1 = EMA(9, 1);
    double emaRapida2 = EMA(9, 2);
    double emaLenta1  = EMA(21, 1);
    double emaLenta2  = EMA(21, 2);
    double p = Close[0]; // precio actual de ejecución

    // Cruce ascendente (señal confirmada)
    if (emaRapida2 < emaLenta2 && emaRapida1 >= emaLenta1) {
        double sl = p - ATR(14, 1) * 1.5;
        double tp = p + ATR(14, 1) * 3.0;
        Buy(1.0, sl, tp);
    }

    // Cruce descendente (señal confirmada)
    if (emaRapida2 > emaLenta2 && emaRapida1 <= emaLenta1) {
        double sl = p + ATR(14, 1) * 1.5;
        double tp = p - ATR(14, 1) * 3.0;
        Sell(1.0, sl, tp);
    }
}
```

9.2) MACD (Divergencia de convergencia de media móvil)

Función	Parámetros	Devolver
`MACD(fast, slow, signal, offset)`	rápido=período EMA rápido, lento=período EMA lento, señal=período de señal, compensación=desplazamiento (0=barra de formación)	Línea MACD (EMA rápida - EMA lenta)
`MACDSignal(fast, slow, signal, offset)`	Mismos parámetros	Línea de señal (MACD EMA)
`MACDHist(fast, slow, signal, offset)`	Mismos parámetros	Histograma (MACD − Señal)

Aviso: el parámetro `offset` es opcional (predeterminado 0). Las tres funciones comparten el mismo cálculo previo: la primera llamada calcula todo, las demás leen del caché.

CRUCE MACD CLÁSICO

```
void OnBarOpen() {
  if (PositionsTotal() > 0) return;

  double hist0 = MACDHist(12, 26, 9, 0);
  double hist1 = MACDHist(12, 26, 9, 1);

  double p = Close[0];
  if (hist1 < 0 && hist0 >= 0) {
    Buy(1.0, p - 100, p + 200);
  }
  if (hist1 > 0 && hist0 <= 0) {
    Sell(1.0, p + 100, p - 200);
  }
}
```

9.3) Bandas de Bollinger

Función	Parámetros	Devolver
`BollingerUpper(period, mult, offset)`	period=período SMA, mult=multiplicador de desviación (ej: 2.0), compensación=desplazamiento	banda superior
`BollingerLower(period, mult, offset)`	Mismos parámetros	banda inferior

La banda media es simplemente `SMA(period, offset)`. El multiplicador acepta valores decimales (1,5, 2,0, 2,5, etc.).

REVERSIÓN EN LAS BANDAS DE BOLLINGER

```
input int inpBBPeriod = 20;
input double inpBBMult = 2.0;

void OnBarOpen() {
  if (PositionsTotal() > 0) return;

  double upper = BollingerUpper(inpBBPeriod, inpBBMult, 0);
  double lower = BollingerLower(inpBBPeriod, inpBBMult, 0);
  double p = Close[0];

  if (p <= lower) {
    Buy(1.0, p - 80, p + 160);
  }
  if (p >= upper) {
    Sell(1.0, p + 80, p - 160);
  }
}
```

9.4) Estocástico (Oscilador estocástico)

Función	Parámetros	Devolver
`StochK(kPeriod, dPeriod, offset)`	kPeriod=ventana %K, dPeriod=suavizado %D, offset=desplazamiento	Línea %K (0–100)
`StochD(kPeriod, dPeriod, offset)`	Mismos parámetros	Línea %D (SMA de %K)

ENTRADA DE SOBREVENTA ESTOCÁSTICA

```
void OnBarOpen() {
  if (PositionsTotal() > 0) return;

  double k0 = StochK(14, 3, 0);
  double d0 = StochD(14, 3, 0);
  double k1 = StochK(14, 3, 1);

  double p = Close[0];
  // Comprar: %K cruza %D hacia arriba desde sobreventa
  if (k1 < d0 && k0 >= d0 && k0 < 30) {
    Buy(1.0, p - 90, p + 180);
  }
}
```

9.5) Canales Keltner

Función	Parámetros	Devolver
`KeltnerUpper(emaPeriod, atrPeriod, mult, offset)`	emaPeriod=EMA del precio, atrPeriod=ATR, mult=multiplicador, offset=desplazamiento	Banda superior (EMA + mult×ATR)
`KeltnerLower(emaPeriod, atrPeriod, mult, offset)`	Mismos parámetros	Banda inferior (EMA − mult×ATR)

La línea central es `EMA(emaPeriod, offset)`. El multiplicador acepta decimales.

APRETÓN DE BOLLINGER (BOLLINGER DENTRO DE KELTNER)

```
void OnBarOpen() {
  double bbUpper = BollingerUpper(20, 2.0, 0);
  double bbLower = BollingerLower(20, 2.0, 0);
  double ktUpper = KeltnerUpper(20, 10, 1.5, 0);
  double ktLower = KeltnerLower(20, 10, 1.5, 0);

  // Squeeze: bandas de Bollinger DENTRO de las bandas de Keltner
  bool squeeze = (bbUpper < ktUpper) && (bbLower > ktLower);

  if (squeeze && PositionsTotal() == 0) {
    double hist = MACDHist(12, 26, 9, 0);
    double p = Close[0];
    if (hist > 0) Buy(1.0, p - 100, p + 200);
    if (hist < 0) Sell(1.0, p + 100, p - 200);
  }
}
```

9.6) Variantes de indicadores de múltiples plazos

Todos los indicadores tienen variantes MTF con el sufijo `TF`, que reciben el marco temporal como primer parámetro:

función local	Equivalente MTF
`SMA(21,0)`	`SMATF(PERIOD_H1,21,0)`
`EMA(21,0)`	`EMATF(PERIOD_H1,21,0)`
`RSI(14,0)`	`RSITF(PERIOD_H1,14,0)`
`ATR(14,0)`	`ATRTF(PERIOD_H1,14,0)`
`MACD(12,26,9,0)`	`MACDTF(PERIOD_H1,12,26,9,0)`
`MACDSignal(12,26,9,0)`	`MACDSignalTF(PERIOD_H1,12,26,9,0)`
`MACDHist(12,26,9,0)`	`MACDHistTF(PERIOD_H1,12,26,9,0)`
`BollingerUpper(20,2.0,0)`	`BollingerUpperTF(PERIOD_H1,20,2.0,0)`
`BollingerLower(20,2.0,0)`	`BollingerLowerTF(PERIOD_H1,20,2.0,0)`
`StochK(14,3,0)`	`StochKTF(PERIOD_H1,14,3,0)`
`StochD(14,3,0)`	`StochDTF(PERIOD_H1,14,3,0)`
`KeltnerUpper(20,10,1.5,0)`	`KeltnerUpperTF(PERIOD_H1,20,10,1.5,0)`
`KeltnerLower(20,10,1.5,0)`	`KeltnerLowerTF(PERIOD_H1,20,10,1.5,0)`

---

10) Comercio: Órdenes, Posiciones y Tickets

Grupo	Funciones principales
Apertura	`Buy`, `Sell`, `BuyLimit`, `SellLimit`, `BuyStop`, `SellStop`
Gestión	`PositionModify`, `OrderModify`, `ClosePosition`, `CloseAll`, `OrderDelete`
Consulta de posición	`PositionsTotal`, `PositionTicket`, `PositionEntry`, `PositionSL`, `PositionTP`, `PositionLots`, `PositionProfit`, `PositionType`
Consulta de pedido	`OrdersTotal`, `OrderTicket`, `OrderPrice`, `OrderSL`, `OrderTP`, `OrderLots`, `OrderType`

11) Bucles y métricas de posición/orden (avanzado)

Esta sección cubre exactamente los patrones de bucle para recorrer órdenes/posiciones abiertas y extraer métricas de gestión útiles.

11.1) Bucle de posición con PnL total y exposición

SUMA POR BUCLE

```
void OnBarOpenM1() {
  double pnlTotal = 0.0;
  double lotsTotal = 0.0;

  for (int i = 0; i < PositionsTotal(); i++) {
    double tk = PositionTicket(i);
    pnlTotal += PositionProfit(tk);
    lotsTotal += PositionLots(tk);
  }

  Print("PnL=" + pnlTotal + "| Lotes=" + lotsTotal);
}
```

11.2) Precio medio ponderado de entrada (todas las posiciones)

PRECIO MEDIO POR LOTE

```
double PrecoMedioPosicoes() {
  double somaPxLots = 0.0;
  double somaLots = 0.0;

  for (int i = 0; i < PositionsTotal(); i++) {
    double tk = PositionTicket(i);
    double lots = PositionLots(tk);
    double px = PositionEntry(tk);

    somaPxLots += px * lots;
    somaLots += lots;
  }

  if (somaLots <= 0) return 0.0;
  return somaPxLots / somaLots;
}

void OnBarOpenM1() {
  if (PositionsTotal() > 0) {
    double pm = PrecoMedioPosicoes();
    Print("Precio medio de las carteras=" + pm);
  }
}
```

11.3) Precio medio separado por dirección (compra/venta)

PROMEDIOS POR LADO

```
void MediasPorLado(double &pmBuy, double &pmSell, double &lotsBuy, double &lotsSell) {
  double somaBuy = 0.0;
  double somaSell = 0.0;
  lotsBuy = 0.0;
  lotsSell = 0.0;

  for (int i = 0; i < PositionsTotal(); i++) {
    double tk = PositionTicket(i);
    double tp = PositionType(tk);   // 0 compra, 1 venta
    double lt = PositionLots(tk);
    double pe = PositionEntry(tk);

    if (tp == 0) {
      somaBuy += pe * lt;
      lotsBuy += lt;
    } else {
      somaSell += pe * lt;
      lotsSell += lt;
    }
  }

  pmBuy = (lotsBuy > 0) ? (somaBuy / lotsBuy) : 0.0;
  pmSell = (lotsSell > 0) ? (somaSell / lotsSell) : 0.0;
}
```

11.4) Punto de equilibrio del lote (ajuste de SL por ticket)

BUCLE CON POSITIONMODIFY

```
void OnBarOpenM1() {
  for (int i = 0; i < PositionsTotal(); i++) {
    double tk = PositionTicket(i);
    double lucro = PositionProfit(tk);

    if (lucro >= 120.0) {
      double entrada = PositionEntry(tk);
      double tp = PositionTP(tk);
      PositionModify(tk, entrada, tp); // lleva a SL al punto de equilibrio
    }
  }
}
```

11.5) Bucle de órdenes pendientes y precio medio de activación

PROMEDIO DE ACTIVACIÓN POR PENDIENTE

```
void OnBarOpenM1() {
  double somaPx = 0.0;
  double somaLots = 0.0;

  for (int i = 0; i < OrdersTotal(); i++) {
    double tk = OrderTicket(i);
    double px = OrderPrice(tk);
    double lt = OrderLots(tk);

    somaPx += px * lt;
    somaLots += lt;
  }

  if (somaLots > 0) {
    double precoMedioPendentes = somaPx / somaLots;
    Print("Precio medio pendiente=" + precoMedioPendentes);
  }
}
```

11.6) Cancelar emisiones pendientes fuera del precio actual

HIGIENE DEL LIBRO DE PEDIDOS

```
input double DistMaxTicks = 120.0;

void OnBarOpenM1() {
  double p = Close[0];
  double lim = DistMaxTicks * TickSize();

  for (int i = OrdersTotal() - 1; i >= 0; i--) {
    double tk = OrderTicket(i);
    double op = OrderPrice(tk);

    if (Abs(op - p) > lim) {
      OrderDelete(tk);
    }
  }
}
```

11.7) Cierre agregado por objetivo de cartera

CERRARTODO POR CONSOLIDATED PNL

```
input double AlvoCarteira = 500.0;
input double StopCarteira = -300.0;

void OnBarOpenM1() {
  double pnl = 0.0;

  for (int i = 0; i < PositionsTotal(); i++) {
    double tk = PositionTicket(i);
    pnl += PositionProfit(tk);
  }

  if (pnl >= AlvoCarteira || pnl <= StopCarteira) {
    CloseAll();
  }
}
```

Nota importante: Al eliminar elementos en un bucle de orden/posición, prefiera iterar hacia atrás (`for i = total-1; i >= 0; i--`) para evitar la inconsistencia del índice después de las eliminaciones.

---

12) Criterio Backtest, Spread y SL-first

M1 OHLC

- Granularidad por vela M1
- Más fieles para la validación
- TP/SL y pendientes valorados continuamente en M1 cuando hay posiciones/órdenes

Plazo de entrada

- Más rápido para la optimización
- Menos detalle intrabar
- Uso para detección; validación final en M1 OHLC

Simulación de propagación: en el backtest, es posible configurar el spread en puntos. Este costo afecta las entradas/salidas según la lógica Bid/Ask simulada.

SL-primer criterio: Cuando se pueden reproducir SL y TP en la misma barra, el motor asume SL primero (postura conservadora).

13) MQR x MT5 (lo que resuelve el conversor)

Punto	MQR	MT5	Estado
indexación de barras	`Close[0]` = forming bar	`iClose(0)` = forming bar	Idéntico (sin ajuste)
División	Semántica double	`int/int` puede truncar	El convertidor ajusta
Módulo	Compatible con double	`%` es solo entero	El convertidor utiliza `fmod`
Eventos	OnBarOpenM1 + OnBarOpen	en tic	El convertidor monta un flujo equivalente
Precio de ejecución	Simulación de motor	Oferta/demanda real del probador	diferencia inherente

14) Lista de verificación de robustez

- Separe la entrada (`OnBarOpen`) de la administración (`OnBarOpenM1`)
- Controlar el riesgo por ticket y cartera (PnL agregado)
- Medir el precio medio por lote y la exposición total
- Borrar pendiente antiguo/lejos del precio actual
- Optimizar en modo rápido y validar en M1 OHLC
- Compare los resultados de MQR x MT5 considerando las diferencias de ejecución inherentes

15) Apéndice: completar ENUM_TIMEFRAMES

Constante	Minutos
PERIODO_ACTUAL	0
PERIODO_M1	1
PERIODO_M2	2
PERIODO_M3	3
PERIODO_M4	4
PERIODO_M5	5
PERIODO_M6	6
PERIODO_M10	10
PERIODO_M12	12
PERIODO_M15	15
PERIODO_M20	20
PERIODO_M30	30
PERIODO_H1	60
PERIODO_H2	120
PERIODO_H3	180
PERIODO_H4	240
PERIODO_H6	360
PERIODO_H8	480
PERIODO_H12	720
PERIODO_D1	1440
PERIODO_W1	10080
PERIODO_MN1	43200

16) Apéndice: Constantes de posición y tipo de orden

POSICIÓN_TYPE_*

Constante	Valor	Descripción
POSITION_TYPE_BUY	0	Posición de compra
POSITION_TYPE_SELL	1	Posición de ventas

ORDER_TYPE_*

Constante	Valor	Descripción
ORDER_TYPE_BUY	0	orden de compra de mercado
ORDER_TYPE_SELL	1	orden de venta de mercado
ORDER_TYPE_BUY_LIMIT	2	Comprar orden límite
ORDER_TYPE_SELL_LIMIT	3	Orden límite de venta
ORDER_TYPE_BUY_STOP	4	Comprar orden de parada
ORDER_TYPE_SELL_STOP	5	Orden de parada de venta

Consejo: Utilice estas constantes al comparar el retorno de `PositionType()` y `OrderType()` en lugar de números fijos.