MQR - Manual Completo

Referência prática da linguagem MQR com foco em backtest, eventos, ordens, posições, loops de gestão e exportação para MT5.

Responsável: Vinicius Fávero
Site: rikdomchart.com.br

Conteúdo

---

1) Visão Geral

MQR é uma DSL para estratégias de trading com sintaxe inspirada em C++/MQL5. O fluxo padrão é: escrever estratégia, rodar backtest, ajustar parâmetros e, quando necessário, exportar para MT5.

Ponto-chave: mesmo sem `OnBarOpenM1()`, o motor mantém precisão de execução de TP/SL e pendentes em M1 quando há posições/ordens abertas no modo M1 OHLC.

2) Arquivos e Pipeline

Extensão	Finalidade
`.mqr`	Código-fonte da estratégia
`.exr`	Estratégia compilada para execução pela engine.

PIPELINE RECOMENDADO

```
1) Escrever a estratégia em MQR
2) Validar no backtest interno
3) Otimizar parâmetros
4) Revalidar em M1 OHLC
5) Exportar para MT5 quando fizer sentido
```

3) Estrutura do Programa e Eventos

Evento	Quando dispara	Uso comum
`OnInit()`	Uma vez no início	SetTimeFrame, estado inicial, buffers
`OnBarOpenM1()`	Fechamento de cada M1	Gestão, horário, trailing, limpeza
`OnBarOpen()`	Primeiro tick após o boundary do TF	Sinais de entrada e filtros

Aliases: `OnBarOpen()` é alias de `OnBarOpen()`, e `OnBarOpenM1()` é alias de `OnBarOpenM1()`. `OnBar()` também funciona como alias de `OnBarOpen()`. Todos produzem o mesmo comportamento.

Semântica MT5-compatível: `OnBarOpen()` dispara no **primeiro tick do candle seguinte** ao boundary, como no MT5. `Close[0]` = barra em formação (forming bar), `Close[1]` = barra completada. Para sinais confirmados, use offset 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 completada, EMA(21,1) = indicador da barra completada
  double ema = EMA(21, 1);
  if (Close[1] > ema) {
    // Close[0] = preco atual (forming bar) para execucao
    Buy(1.0, Close[0] - 80, Close[0] + 120);
  }
}
```

---

4) Tipos, Inputs, Enums e Operadores

Tipos base

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

Operadores

- Aritméticos: `+ - * / %`
- Comparação: `== != < > <= >=`
- Lógicos: `&& || !`
- Atribuição: `= += -= *= /= ++ --`

ENUM + INPUTS

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

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

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

---

5) Funções Matemáticas

A linguagem MQR oferece funções matemáticas built-in que operam sobre valores `double`. São usadas para arredondamento, limites, potências e cálculos gerais.

Função	Parâmetros	Retorno	Descrição
`Abs(x)`	x: double	double	Valor absoluto de x
`Max(x, y)`	x, y: double	double	Maior valor entre x e y
`Min(x, y)`	x, y: double	double	Menor valor entre x e y
`Sqrt(x)`	x: double	double	Raiz quadrada de x
`Round(x)`	x: double	double	Arredonda para o inteiro mais próximo
`Floor(x)`	x: double	double	Arredonda para baixo (maior inteiro ≤ x)
`Ceil(x)`	x: double	double	Arredonda para cima (menor inteiro ≥ x)
`Pow(base, exp)`	base, exp: double	double	Potência: base elevado a exp

EXEMPLOS DE USO

```
// Arredondamento de horário
int horaFechamento = Floor(inpHora / 100);
int minutoFechamento = inpHora % 100;

// Stop dinâmico com ATR arredondado
double stopPts = Round(ATR(14, 0) * 1.5);

// Tamanho do range em ticks
double rangeTicks = Round(Abs(High[0] - Low[0]) / TickSize());

// Maior high dos últimos 3 candles
double maxHigh = Max(High[0], Max(High[1], High[2]));

// Potência para cálculos de desvio padrão manual
double diff = Close[0] - SMA(20, 0);
double diffSq = Pow(diff, 2.0);

// Raiz quadrada (ex: para normalização)
double vol = Sqrt(Pow(High[0] - Low[0], 2.0));
```

---

6) Funções de Data e Hora

Funções para consultar hora, minuto e dia da semana do candle M1 corrente. Úteis para filtros de horário de operação, encerramentos e controle de sessão.

Função	Retorno	Descrição
`Hour()`	int	Hora do candle M1 atual (0–23)
`Minute()`	int	Minuto do candle M1 atual (0–59)
`DayOfWeek()`	int	Dia da semana (0=Domingo, 1=Segunda, …, 6=Sábado)
`IsFirstBarOfDay()`	bool	True se é o primeiro candle do dia no timeframe principal
`IsLastBarOfDay()`	bool	True se é o último candle do dia no timeframe principal
`IsFirstBarOfDayTF(tf)`	bool	Mesma lógica, mas no timeframe `tf`
`IsLastBarOfDayTF(tf)`	bool	Mesma lógica, mas no timeframe `tf`

FILTRO DE HORÁRIO PADRÃO

```
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() {
    // Fecha tudo às 17:30
    if (PositionsTotal() > 0 && (Hour() > 17 || (Hour() == 17 && Minute() >= 30))) {
        CloseAll();
    }
}

void OnBarOpen() {
    // Reset de estado no primeiro candle do dia
    if (IsFirstBarOfDay()) {
        // ... reiniciar variáveis
    }
}
```

EXEMPLO PRÁTICO VALIDAÇÃO DE HORÁRIOS: INÍCIO, LIMITE DE ENTRADA E FECHAMENTO TOTAL

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

  input ENUM_HORA inpHoraInicio = H0905; // pode abrir novas ordens
  input ENUM_HORA inpHoraLimite = H1630; // após isso, nao abre novas ordens
  input ENUM_HORA inpHoraFech   = H1720; // fechamento 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() {
    // Garante fechamento no horario exato (granularidade M1)
    FecharPorHorario();
  }

  void OnBarOpen() {
    // Tambem protege no fluxo principal
    FecharPorHorario();

    // Sem novas entradas apos o horario limite
    if (!DentroDoHorarioOrdens()) return;

    // ... logica de entrada
  }
```

7) Propriedades do Ativo, Arrays e Utilitários

7.1) Propriedades do ativo

Informações do instrumento financeiro corrente, úteis para calcular stops em pontos, normalizar preços e dimensionar lotes.

Função	Retorno	Descrição
`TickSize()`	double	Menor variação de preço do ativo (ex: 0.01 para mini-índice, 0.5 para mini-dólar)
`TickValue()`	double	Valor monetário de 1 tick por lote (ex: R$0.20 para WINFUT)
`Digits()`	int	Número de casas decimais do preço
`NormalizePrice(price)`	double	Arredonda o preço para o incremento válido do ativo

STOP BASEADO EM TICKS E PREÇO 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) Arrays

Suporte a arrays dinâmicos para armazenar séries e coleções de valores.

Função	Parâmetros	Descrição
`ArrayResize(arr, size)`	arr: array, size: int	Redimensiona o array para o novo tamanho
`ArraySize(arr)`	arr: array	Retorna o tamanho atual do array

EXEMPLO COM ARRAY

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

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

    if (ArraySize(historico) >= 10) {
        // usar os últimos 10 valores
    }
}
```

7.3) Print (debug)

Imprime valores no log de execução para depuração durante backtest.

Função	Parâmetros	Descrição
`Print(value)`	value: double ou string	Imprime no log de execução. Use concatenação com + para compor mensagens.

DEBUG COM PRINT

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

---

8) Timeframes e Multi-Timeframe (MTF)

O timeframe principal é definido em `OnInit()` com `SetTimeFrame()`. Leituras MTF usam funções `*TF` sem perder o contexto do candle principal.

8.1) Dados do timeframe do backtest (contexto principal)

As séries `Open[]`, `High[]`, `Low[]`, `Close[]`, `Volume[]` e `Time[]` sempre refletem o timeframe principal configurado na estratégia. Esse é o contexto onde `OnBarOpen()` toma decisão de entrada.

Função/Série	Descrição
`Open[n], High[n], Low[n], Close[n]`	OHLC do timeframe principal (n=0 barra em formação, n=1 última completada)
`Volume[n], Time[n]`	Volume e timestamp do candle do contexto principal
`BarIndex(), BarCount()`	Índice atual e quantidade de barras carregadas

LEITURA DO CONTEXTO PRINCIPAL

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

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

  // Exemplo: aceleração simples na barra completada
  if (c1 > c2 && rangeCompletada > TickSize() * 20) {
    Print("Sinal no TF principal");
  }
}
```

8.2) Dados de outros timeframes

Para consultar contexto externo (H1, H4, D1 etc.), use família `*TF`: `OpenTF`, `EMATF`, `RSITF`, etc. Isso evita reconstruir candles manualmente.

Categoria	Funções
OHLC/volume/tempo	`OpenTF`, `HighTF`, `LowTF`, `VolumeTF`, `TimeTF`
Indicadores	`SMATF`, `EMATF`, `RSITF`, `ATRTF`, `MACDTF`, `MACDSignalTF`, `MACDHistTF`, `BollingerUpperTF`, `BollingerLowerTF`, `StochKTF`, `StochDTF`, `KeltnerUpperTF`, `KeltnerLowerTF`
Estado da barra	`BarCountTF`, `IsBarCompleteTF`, `IsNewBarTF`

ENTRADA EM M5 COM FILTRO ESTRUTURAL DE H1

```
void OnBarOpen() {
  // Contexto do backtest (M5, por exemplo)
  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) Como trabalhar com indicadores no mesmo código

Uma abordagem prática é separar: contexto local para gatilho e contexto externo para direção. Exemplos comuns:

- Direção macro em H1/H4 com `EMATF`.
- Timing de entrada no TF principal com `RSI` ou cruzamento de médias locais.
- Dimensionamento de stop por volatilidade com `ATR` (local) ou `ATRTF` (externo).

COMBINAÇÃO PRÁTICA: TENDÊNCIA H1 + TIMING LOCAL + STOP POR 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) Diferença prática: dado local vs dado externo

Regra rápida: `Close[0]` é a **barra em formação** (forming bar) do timeframe da estratégia — mesmo conceito do MT5. `Close[1]` é a última barra completada. `OpenTF(PERIOD_H1, 0)` retorna o Open do H1, mesmo que sua estratégia rode em M1/M5.

FILTRO DE TENDÊNCIA H1 EM ESTRATÉGIA 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) Confluência tripla de timeframe (M5 + H1 + H4)

Exemplo de entrada local no timeframe do backtest com dupla confirmação externa. Esse padrão reduz sinais contra a direção macro.

CONFLUÊNCIA DE TENDÊNCIA EM TRÊS HORIZONTES

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

  double p = Close[0];                  // TF do backtest
  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 com RSI externo (evitar entrada em exaustão)

Neste padrão, o gatilho acontece no TF principal, mas o RSI de H1 valida se o movimento externo já está esticado.

RSI LOCAL + RSI H1

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

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

  // Exemplo de leitura: compra apenas com pullback local e H1 ainda saudável
  if (rsiM5 < 35 && rsiH1 > 45 && rsiH1 < 70) {
    Buy(1.0, p - 90, p + 170);
  }
}
```

8.7) Sincronização por nova barra de outro timeframe

Use `IsNewBarTF()` para atualizar variáveis de contexto somente quando fechar barra do TF externo, evitando recalcular tudo em todo candle local.

ATUALIZAÇÃO DE CONTEXTO H1 APENAS NO FECHAMENTO 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 = acima da média
  }
}

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 stop dinâmico

Quando o TF principal é muito curto, usar ATR de H1/H4 pode estabilizar o tamanho de stop em cenários de ruído intraday.

STOP BASEADO EM ATR DE H1

```
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) Exemplo de leitura combinada de OHL em TF externo

Exemplo simples de amplitude de candle em H1 usado como filtro para entrada local no timeframe do backtest. Usamos apenas Open, High e Low do TF externo.

FILTRO DE AMPLITUDE 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;

  // Abertura na metade inferior sugere pressão compradora
  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

A linguagem MQR oferece indicadores técnicos built-in com precomputação lazy e lookup O(1) por barra. Todos aceitam um parâmetro `offset` opcional (padrão 0), onde 0 = barra em formação (forming bar), 1 = última barra completada, etc.

9.1) Indicadores Básicos

Função	Parâmetros	Retorno	Descrição
`SMA(period, offset)`	period=número de barras, offset=deslocamento (0=forming bar)	double	Média Móvel Simples
`EMA(period, offset)`	period=número de barras, offset=deslocamento	double	Média Móvel Exponencial
`RSI(period, offset)`	period=número de barras, offset=deslocamento	double (0–100)	Índice de Força Relativa
`ATR(period, offset)`	period=número de barras, offset=deslocamento	double	Average True Range (volatilidade)

CRUZAMENTO DE MÉDIAS CLÁSSICO

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

    // Barras completadas: offset 1 e 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]; // preco atual para execucao

    // Cruzamento para cima (sinal confirmado)
    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);
    }

    // Cruzamento para baixo (sinal confirmado)
    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 (Moving Average Convergence Divergence)

Função	Parâmetros	Retorno
`MACD(fast, slow, signal, offset)`	fast=período EMA rápida, slow=período EMA lenta, signal=período sinal, offset=deslocamento (0=forming bar)	Linha MACD (EMA rápida − EMA lenta)
`MACDSignal(fast, slow, signal, offset)`	Mesmos parâmetros	Linha de sinal (EMA do MACD)
`MACDHist(fast, slow, signal, offset)`	Mesmos parâmetros	Histograma (MACD − Signal)

Nota: o parâmetro `offset` é opcional (padrão 0). As três funções compartilham a mesma precomputação — a primeira chamada calcula tudo, as demais leem do cache.

CRUZAMENTO MACD CLÁSSICO

```
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

Função	Parâmetros	Retorno
`BollingerUpper(period, mult, offset)`	period=período SMA, mult=multiplicador desvio (ex: 2.0), offset=deslocamento	Banda superior
`BollingerLower(period, mult, offset)`	Mesmos parâmetros	Banda inferior

A banda do meio é simplesmente `SMA(period, offset)`. O multiplicador aceita valores decimais (1.5, 2.0, 2.5 etc).

REVERSÃO NAS 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 (Stochastic Oscillator)

Função	Parâmetros	Retorno
`StochK(kPeriod, dPeriod, offset)`	kPeriod=janela %K, dPeriod=suavização %D, offset=deslocamento	Linha %K (0–100)
`StochD(kPeriod, dPeriod, offset)`	Mesmos parâmetros	Linha %D (SMA de %K)

ENTRADA POR SOBREVENDA DO ESTOCÁSTICO

```
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];
  // Compra: %K cruza %D para cima vindo de sobrevenda
  if (k1 < d0 && k0 >= d0 && k0 < 30) {
    Buy(1.0, p - 90, p + 180);
  }
}
```

9.5) Canais de Keltner (Keltner Channels)

Função	Parâmetros	Retorno
`KeltnerUpper(emaPeriod, atrPeriod, mult, offset)`	emaPeriod=EMA do preço, atrPeriod=ATR, mult=multiplicador, offset=deslocamento	Banda superior (EMA + mult×ATR)
`KeltnerLower(emaPeriod, atrPeriod, mult, offset)`	Mesmos parâmetros	Banda inferior (EMA − mult×ATR)

A linha central é `EMA(emaPeriod, offset)`. O multiplicador aceita decimais.

BOLLINGER SQUEEZE (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 das 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 Multi-Timeframe dos Indicadores

Todos os indicadores possuem variantes MTF com sufixo `TF`, que recebem o timeframe como primeiro parâmetro:

Função 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) Trade: Ordens, Posições e Tickets

Grupo	Funções principais
Abertura	`Buy`, `Sell`, `BuyLimit`, `SellLimit`, `BuyStop`, `SellStop`
Gestão	`PositionModify`, `OrderModify`, `ClosePosition`, `CloseAll`, `OrderDelete`
Consulta posição	`PositionsTotal`, `PositionTicket`, `PositionEntry`, `PositionSL`, `PositionTP`, `PositionLots`, `PositionProfit`, `PositionType`
Consulta ordem	`OrdersTotal`, `OrderTicket`, `OrderPrice`, `OrderSL`, `OrderTP`, `OrderLots`, `OrderType`

11) Loops e Métricas de Posição/Ordem (avançado)

Esta seção cobre exatamente os padrões de loop para percorrer ordens/posições abertas e extrair métricas úteis de gestão.

11.1) Loop de posições com PnL total e exposição

SOMATÓRIO POR LOOP

```
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 + " | Lots=" + lotsTotal);
}
```

11.2) Preço médio ponderado de entrada (todas as posições)

PREÇO MÉDIO POR LOTES

```
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("Preco medio carteiras=" + pm);
  }
}
```

11.3) Preço médio separado por direção (compra/venda)

MÉDIAS 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 venda
    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) Break-even em lote (ajuste de SL por ticket)

LOOP COM 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); // move SL para break-even
    }
  }
}
```

11.5) Loop de ordens pendentes e preço médio de ativação

MÉDIA DE GATILHO POR PENDENTES

```
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("Preco medio pendentes=" + precoMedioPendentes);
  }
}
```

11.6) Cancelar pendentes longe do preço atual

HIGIENE DE BOOK DE ORDENS

```
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) Fechamento agregado por alvo de carteira

CLOSEALL POR PNL CONSOLIDADO

```
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();
  }
}
```

Observação importante: quando for remover itens em loop de ordens/posições, prefira iterar de trás para frente (`for i = total-1; i >= 0; i--`) para evitar inconsistência de índice após exclusões.

---

12) Backtest, Spread e Critério SL-first

M1 OHLC

- Granularidade por candle M1
- Mais fiel para validação
- TP/SL e pendentes avaliados continuamente em M1 quando há posições/ordens

Timeframe do input

- Mais rápido para otimização
- Menos detalhe intrabar
- Use para triagem; valide final em M1 OHLC

Simulação de spread: no backtest, é possível configurar spread em pontos. Esse custo afeta entradas/saídas conforme a lógica Bid/Ask simulada.

Critério SL-first: quando SL e TP poderiam ser tocados na mesma barra, a engine assume SL primeiro (postura conservadora).

13) MQR x MT5 (o que o conversor resolve)

Ponto	MQR	MT5	Status
Indexação de barras	`Close[0]` = forming bar	`iClose(0)` = forming bar	Idêntica (sem ajuste)
Divisão	Semântica em double	`int/int` pode truncar	Conversor ajusta
Módulo	Compatível com double	`%` é inteiro	Conversor usa `fmod`
Eventos	OnBarOpenM1 + OnBarOpen	OnTick	Conversor monta fluxo equivalente
Preço de execução	Simulação da engine	Bid/Ask real do tester	Diferença inerente

14) Checklist de Robustez

- Separar entrada (`OnBarOpen`) de gestão (`OnBarOpenM1`)
- Controlar risco por ticket e por carteira (PnL agregado)
- Medir preço médio por lote e exposição total
- Limpar pendentes antigas/longe do preço atual
- Otimizar em modo rápido e validar em M1 OHLC
- Comparar resultados MQR x MT5 considerando diferenças inerentes de execução

15) Apêndice: ENUM_TIMEFRAMES completo

Constante	Minutos
PERIOD_CURRENT	0
PERIOD_M1	1
PERIOD_M2	2
PERIOD_M3	3
PERIOD_M4	4
PERIOD_M5	5
PERIOD_M6	6
PERIOD_M10	10
PERIOD_M12	12
PERIOD_M15	15
PERIOD_M20	20
PERIOD_M30	30
PERIOD_H1	60
PERIOD_H2	120
PERIOD_H3	180
PERIOD_H4	240
PERIOD_H6	360
PERIOD_H8	480
PERIOD_H12	720
PERIOD_D1	1440
PERIOD_W1	10080
PERIOD_MN1	43200

16) Apendice: Constantes de Tipo de Posicao e Ordem

POSITION_TYPE_*

Constante	Valor	Descricao
POSITION_TYPE_BUY	0	Posicao de compra
POSITION_TYPE_SELL	1	Posicao de venda

ORDER_TYPE_*

Constante	Valor	Descricao
ORDER_TYPE_BUY	0	Ordem de compra a mercado
ORDER_TYPE_SELL	1	Ordem de venda a mercado
ORDER_TYPE_BUY_LIMIT	2	Ordem Buy Limit
ORDER_TYPE_SELL_LIMIT	3	Ordem Sell Limit
ORDER_TYPE_BUY_STOP	4	Ordem Buy Stop
ORDER_TYPE_SELL_STOP	5	Ordem Sell Stop

Dica: Use essas constantes ao comparar o retorno de `PositionType()` e `OrderType()` em vez de numeros fixos.